Specification and Verification of Multi-Agent Systems

Wojciech Jamroga[1] and Wojciech Penczek[2]

[1] Computer Science and Communication, University of Luxembourg
wojtek.jamroga@uni.lu
[2] Institute of Computer Science, PAS, and University of Natural Science and Humanities, Poland
penczek@ipipan.waw.pl

1 Introduction

Overview. Multi agent systems (MAS) provide an important framework for formalizing various problems in computer science, artificial intelligence, game theory, social choice theory, etc. Modal logics are amongst the most suitable and versatile logical formalisms for specification and verification of computational systems. Here, we present an overview of some important developments in the area. We introduce modal logics used for specification of temporal, epistemic, and strategic properties of systems; then, we present some model checking algorithms, and discuss the computational complexity of the model checking problem. Finally, we consider symbolic (compact) representations of systems based on Binary Decision Diagrams (BDD) and propositional logic formulas, and show how the representations change the algorithmic side of model checking. We also discuss other techniques that help to reduce the complexity and make the verification feasible even for large systems.

Contents. The materials consist of 5 parts, with the content outlined below:

1. *Reasoning about evolution of systems.* Multi-agent systems. Modal logic, Kripke models, epistemic logic. Temporal logic, linear vs. branching time, synchronous vs. asynchronous product; reasoning about knowledge and time; μ-calculus. Basic complexity classes, basic complexity results.
2. *Introduction to model checking for knowledge and time.* Standard non-symbolic algorithms. Introduction to symbolic model checking.
3. *Specification of agents and their teams.* Strategic logics: Coalition Logic, ATL. Meta-properties: axiomatization, model equivalence. Reasoning about scenarios with imperfect information.
4. *Complexity of verification.* Model checking complexity for explicit models, complexity proofs. State-space explosion. Complexity revisited: compact representation of transitions, higher-order representations.
5. *Practical model checking.* BDD-based and SAT-based approaches to model checking, bounded and unbounded model checking for CTLK, unbounded model checking for ATL.

Further reading. The following publications may be of particular interest to readers of this chapter:

- R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.
- A. Lomuscio, W. Penczek. Logic Column 19: Symbolic Model Checking for Temporal-Epistemic Logics, CoRR abs/0709.0446, 2007.
- N. Bulling, J. Dix, and W. Jamroga. Model Checking Logics of Strategic Ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pp. 125–159. Springer, 2010.

We give more detailed references for further reading throughout the rest of the materials.

About the authors. Wojciech Jamroga is a research associate at the Individual and Collective Reasoning group at the University of Luxembourg. He works mainly on logical and game-theoretical aspects of multi-agent systems. His teaching record includes five courses at ESSLLI (2006, 2007, 2008, 2010, and 2011), and three courses at EASSS (European Agent Systems Summer School).

Wojciech Penczek is the head of the Theory of Distributed Systems group at ICS PAS, Warsaw. He was a research fellow at the Technical University of Eindhoven and Manchester University; he also worked as a consultant at AT&T. His research is now focused on verification methods for (timed) distributed and multi-agent systems. He has been a project leader of the EC-founded project CRIT-2 and played a main role in several national projects. He has chaired and was a PC member of many conferences on Computer Science. He is the author of more than 100 conference and journal articles on temporal logic, verification methods, model checking, and formal theories of multi-agent systems. His teaching record, besides numerous regular undergraduate courses, includes two courses at EASSS (European Agent Systems Summer School).

Acknowledgements. The materials include fragments that had been prepared together with the following researchers: Thomas Agotnes, Nils Buling, Jürgen Dix, Valentin Goranko, Hongyang Hu, Magdalena Kacprzak, Alessio Lomuscio, Maciej Szreter, and Bożena Woźna-Szcześniak. We gratefully acknowledge their contribution. We also thank the reviewers of this chapter for their insights and useful comments.

Wojciech Jamroga acknowledges the support of the FNR (National Research Fund) Luxembourg under project S-GAMES – C08/IS/03. Wojciech Penczek's work on these materials was partly supported by National Science Center, Poland, under the grant No. 2011/01/B/ST6/01477.

2 Reasoning about Evolution of Systems

This section serves as an introduction of some fundamental concepts that we are going to use throughout.

2.1 Multi-Agent Systems

Multi-agent systems (*MAS*) are systems that involve several autonomous entities acting in the same environment. The entities are called *agents*. What is an agent? Despite numerous attempts to answer this question, there seems to be no conclusive definition. We assume that MAS are, most of all, a philosophical metaphor that induces a specific way of seeing the world, and makes us use agent-oriented vocabulary when describing the phenomena we are interested in. Thus, while some researchers present multi-agent systems as a new paradigm for computation or design, we believe that primarily multi-agent systems form a new paradigm for *thinking* and *talking* about the world, and assigning it a specific conceptual structure. The metaphor of a multi-agent system seems to build on the intuition that *we* are agents – and that other entities we study can be just like us to some extent. The usual properties of agents, like autonomy, pro-activeness etc., seem to be secondary: they are results of an introspection rather than primary assumptions we start with.

We note that this view of multi-agent systems comes close to the role of *logic* in both philosophy and computer science. Logic provides conceptual structures for *modeling* and *reasoning* about the world in a precise manner – and, occasionally, it also provides tools to do it automatically.

References: Reader interested in issues related to multi-agent systems is referred to [105, 104].

2.2 Modal Logic

Modal logic is an extension of classical logic with new operators $\Box$ (*necessity*) and $\Diamond$ (*possibility*): $\Box p$ means that p is true in every possible scenario, while $\Diamond p$ means that p is true in at least one possible scenario. Let $\mathcal{PV}$ be the set of *propositional variables* (also called *propositions*). Models of modal logics are called *Kripke models* or *possible world models*, and include the set of *possible worlds* (or states) St, modal accessibility relation $\mathcal{R} \subseteq St \times St$, and interpretation of the propositions $V : \mathcal{PV} \to 2^{St}$. Now, for a model $M = (St, \mathcal{R}, V)$ and world q in M:

$$M, q \models \Box\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q\mathcal{R}q'$$
$$M, q \models \Diamond\varphi \text{ iff } M, q' \models \varphi \text{ for some } q' \text{ such that } q\mathcal{R}_i q'$$

Modal logic can be further extended to multi-modal logic, where we deal with several modal operators: $\Box_i$ and $\Diamond_i$, each of them interpreted over the corresponding i-modal accessibility relation $\mathcal{R}_i \subseteq St \times St$.

The actual accessibility relations can capture various dimensions of the reality (and therefore give rise to different kinds of modal logics): knowledge ($\rightsquigarrow$ *epistemic logic*), beliefs ($\rightsquigarrow$ *doxastic logic*), obligations ($\rightsquigarrow$ *deontic logic*), actions ($\rightsquigarrow$ *dynamic logic*), time ($\rightsquigarrow$ *temporal logic*) etc. In particular, various aspects of agents (and agent systems) can be naturally captured within this generic framework.

References: A gentle introduction to modal logic can be found in [7].

2.3 Reasoning about Knowledge

The basic *epistemic logic* which we consider here involves modalities for individual agent's knowledge K_i, with $\mathrm{K}_i\varphi$ interpreted as "agent i knows that φ". Additionally, one can consider modalities for collective knowledge of groups of agents: *mutual knowledge* ($\mathrm{E}_A\varphi$: "everybody in group A knows that φ"), *common knowledge* ($\mathrm{C}_A\varphi$: "all the agents in A know that φ, and they know that they know it etc."), and *distributed knowledge* ($\mathrm{D}_A\varphi$: "if the agents could share their individual information, they would be able to recognize that φ"). Note that $\mathrm{E}_A\varphi \equiv \bigwedge_{i\in A} \mathrm{K}_i\varphi$, and hence the operators for mutual knowledge can be omitted from the language.

The formal semantics for the logic is based on *multi-agent epistemic models* of the type $\langle St, \sim_1, ..., \sim_k, V\rangle$, where St is a set of *epistemic states*, V is a valuation of propositions, and each $\sim_i \subseteq St \times St$ is an equivalence relation that defines the *indistinguishability of states* for agent i. Operators K_i are provided with the usual Kripke semantics given by the clause:

$$M, q \models \mathrm{K}_i\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_i q'$$

The accessibility relation corresponding to E_A is defined as $\sim_A^{\mathrm{E}} = \bigcup_{i\in A} \sim_i$, and the semantics of E_A becomes

$$M, q \models \mathrm{E}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{E}} q'.$$

Common knowledge C_A is given semantics in terms of the relation $\sim_A^{\mathrm{C}}$ defined as the transitive closure of $\sim_A^{\mathrm{E}}$:

$$M, q \models \mathrm{C}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{C}} q'.$$

Finally, distributed knowledge $\mathrm{D_A}$ is given semantics in terms of the relation $\sim_A^{\mathrm{D}}$ defined as $\bigcap_{i\in A} \sim_i$, following the same pattern:

$$M, q \models \mathrm{D}_A\varphi \text{ iff } M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_A^{\mathrm{D}} q'.$$

Example 1 (Robots and Carriage). Consider the scenario depicted in Figure 1. Two robots push a carriage from opposite sides. As a result, the carriage can move clockwise or anticlockwise, or it can remain in the same place – depending on who pushes with more force (and, perhaps, who refrains from pushing). To make our model of the domain discrete, we identify 3 different positions of the

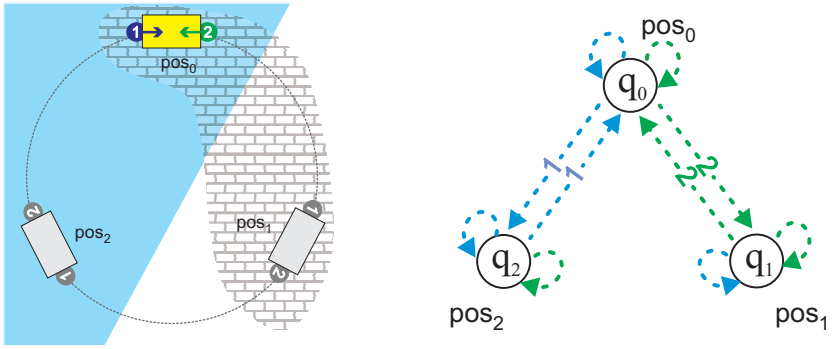

Fig. 1. Two robots and a carriage: a schematic view (left) and a Kripke model M_1 of the robots' knowledge (right).

carriage, and associate them with states q_0, q_1, and q_2. We label the states with propositions $\mathsf{pos}_0, \mathsf{pos}_1, \mathsf{pos}_2$, respectively, to allow for referring to the current position of the carriage in the object language.

Moreover, we assume that robot 1 is only able to observe the color of the surface on which it is standing, and robot 2 perceives only the texture. As a consequence, the first robot can distinguish between position 0 and position 1, but positions 0 and 2 look the same to it. Likewise, the second robot can distinguish between positions 0 and 2, but not 0 and 1. In the resulting epistemic model, we have for instance that $M_1, q_0 \models \neg \mathrm{K}_1 \mathsf{pos}_0 \wedge \neg \mathrm{K}_1 \mathsf{pos}_2 \wedge \mathrm{K}_1(\mathsf{pos}_0 \vee \mathsf{pos}_2)$: the first robot knows that the position is either 0 or 2, but not which of them precisely. Moreover, $M_1, q_0 \models \mathrm{K}_1 \neg \mathsf{pos}_1$ (robot 1 knows that the current position is not 1). The robot also knows that the other agent can distinguish between smooth and rough texture: $M_1, q_0 \models \mathrm{K}_1\big((\mathsf{pos}_2 \to \mathrm{K}_2 \mathsf{pos}_2) \wedge (\neg \mathsf{pos}_2 \to \mathrm{K}_2 \neg \mathsf{pos}_2)\big)$. Finally, if the robots share their information, they know the current position precisely: $M_1, q_0 \models \mathrm{D}_{\{1,2\}} \mathsf{pos}_0$.

Note that epistemic models are usually constructed from the point of view of an *external omniscient observer*, most typically a system designer who has a complete view of the entire system.

References: [28] presents what has become the standard treatment of reasoning about knowledge within the computer science community. More lightweight surveys can be found in [36, 97].

2.4 Modal Logics of Time and Action

We present a brief overview of logics that regard the dynamics of systems. That is, essentially, logics that focus on actions that can be performed by (or in) a system, and on the way in which the system can evolve over time. We consider

two basic cases here: either actions are first-class citizens of the language, or we abstract from them and reason about *change* in general.

PDL. *Propositional Dynamic Logic (PDL)*, which was primarily designed to reason about computer programs, is probably the most typical representative of logics with explicit actions. Actions are represented in the language with *action labels* $\alpha_1, \alpha_2, ...$ from a finite set *Act*. Complex action terms can be also constructed, for example sequential composition $(\alpha_1; \alpha_2)$, nondeterministic choice $(\alpha_1 \cup \alpha_2)$, finite iteration $(\alpha*)$ etc. Now, $[\alpha]\varphi$ expresses the fact that φ is bound to hold after *every* execution of α, and $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$ says that φ holds after *at least one* possible execution of α. On the semantic side, we have *Labeled Transition Systems* $M = \langle St, \stackrel{\alpha_1}{\to}, \stackrel{\alpha_2}{\to}, ..., V\rangle$, where actions are modeled as (nondeterministic) state transformations $\stackrel{\alpha}{\to} \subseteq St \times St$, and the following semantic clause:

$$M, q \models [\alpha]\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \stackrel{\alpha}{\to} q'.$$

We mention PDL only in passing here, as it will not be used in the rest of the chapter.

LTL. *Temporal logics* leave actions implicit, and instead focus on possible patterns of evolution. Typical temporal operators are: X ("in the next state"), G ("always from now on"), F ("sometime in the future"), and U ("until"). Models include one transition relation, and come in two versions. *Linear time* models define a total ordering on possible worlds ("time moments"), so that a model can be seen as a single infinite path λ with successive states $\lambda[0], \lambda[1], ...$; by λ_i we denote the suffix of λ starting from the state $\lambda[i]$. The semantics of *Linear Temporal Logic (LTL)* can be defined as follows:

$$\begin{aligned}
\lambda &\models p \text{ iff } \lambda[0] \in V(p);\\
\lambda &\models \neg\varphi \text{ iff } \lambda \not\models \varphi,\\
\lambda &\models \varphi \wedge \psi \text{ iff } \lambda \models \varphi \text{ and } \lambda \models \psi;\\
\lambda &\models \mathrm{X}\varphi \text{ iff } \lambda_1 \models \varphi;\\
\lambda &\models \varphi \mathrm{U} \psi \text{ iff } (\exists j \geq 0)\big(\lambda_j \models \psi \text{ and } (\forall 0 \leq i < j)\ \lambda_i \models \varphi\big).
\end{aligned}$$

with $\mathrm{F}\varphi \equiv \mathbf{true}\mathrm{U}\varphi$ and $\mathrm{G}\varphi \equiv \neg\mathrm{F}\neg\varphi$, where $\mathbf{true} \stackrel{def}{=} p \vee \neg p$, for some $p \in \mathcal{PV}$.

Example 2. Consider the path $(q_0q_1q_2)^\omega = q_0q_1q_2q_0q_1q_2q_0q_1q_2\ldots$ from the robots and carriage scenario (Example 1). For that path, we have for instance that $\mathrm{F}\mathsf{pos}_2$ (position 2 will be eventually achieved), and even $\mathrm{GF}\mathsf{pos}_2$ (position 2 will be achieved infinitely many times). However, it is not the case that the carriage will stop and stay in position 0 for good: $\neg\mathrm{FG}\mathsf{pos}_2$.

Typically, when LTL is used for specifying properties of a system, the formulae are interpreted over all the infinite paths from a transition model of the system.

CTL*. *Branching-time models*, on the other hand, consist of a tree that encapsulates all possible evolutions of the system. *Computation Tree Logic (CTL*)*

extends LTL with *path quantifiers* E ("there is a path"), and A ("for every path"). Formally, the syntax and semantics of CTL* is defined in the following way. Let $\mathcal{PV} = \{p_1, p_2 \ldots\}$ be a set of propositional variables. The language of CTL* is given as the set of all the state formulas φ (interpreted at states of a model), defined using path formulas γ (interpreted at paths of a model), by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{A}\gamma \mid \mathsf{E}\gamma$$
$$\gamma := \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathrm{X}\gamma \mid \gamma \mathrm{U} \gamma.$$

In the above $p \in \mathcal{PV}$, A ('for All paths') and E ('there Exists a path') are path quantifiers, whereas X ('neXt') and U ('Until') are temporal operators like before.

Obviously, when verifying properties of a particular system, a model given in the form of a set of infinite paths (for LTL) or an infinite tree (CTL*) would be impractical. Instead, the behavior of the system is usually represented as a Kripke model of transitions – either labeled (with multiple transition relations, one per action name) or unlabeled (all transitions "collapsed" into a single relation $\rightarrow$). Therefore, semantics of CTL* is frequently defined in terms of standard Kripke models. Notice that the unfolding of a Kripke model is an inifinite tree.

Let $M = (St, \rightarrow, V)$ be a Kripke model. For $q_0 \in St$ a (full) *path* $\lambda = (q_0, q_1, \ldots)$ is an infinite sequence of states in St starting at q_0, where $q_i \rightarrow q_{i+1}$ for all $i \geq 0$, and $\lambda_i = (q_i, q_{i+1}, \ldots)$ is the i-th suffix of λ starting at q_i. By $M, q \models \varphi$ ($M, \lambda \models \gamma$) we mean that φ holds in the state q (γ holds on the path λ, respectively) of the model M. In what follows the model M is sometimes omitted if it is clear from the context. The relation $\models$ is defined inductively below.

$M, q \models p$ iff $q \in V(p)$, for $p \in \mathcal{PV}$,
$M, x \models \neg y$ iff $M, x \not\models y$, for $x \in \{q, \lambda\}, y \in \{\varphi, \gamma\}$,
$M, x \models y_1 \wedge y_2$ iff $M, x \models y_1$ and $M, x \models y_2$, for x, y as above,
$M, q \models \mathrm{A}\gamma$ iff $M, \lambda \models \gamma$ for each path λ starting at q,
$M, q \models \mathrm{E}\gamma$ iff $M, \lambda \models \gamma$ for some path λ starting at q,
$M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$, for a state formula φ,
$M, \lambda \models \mathrm{X}\gamma$ iff $M, \lambda_1 \models \gamma$,
$M, \lambda \models \gamma_1 \mathrm{U} \gamma_2$ iff $(\exists j \geq 0)\big(M, \lambda_j \models \gamma_2$ and $(\forall 0 \leq i < j)\ M, \lambda_i \models \gamma_1\big)$.

The logic CTL is an important subset of $\mathrm{CTL}^*_{-\mathrm{X}}$. In "pure" (or "vanilla") CTL every occurrence of a path quantifier is followed by exactly one temporal operator, so there are only state formulae. Note that in the case of "vanilla" CTL, there exists an alternative semantics given entirely in terms of satisfaction of formulae in *states*:

$M, q \models p$ iff $q \in V(p)$;
$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;
$M, q \models \mathsf{EX}\varphi$ iff $M, \lambda[1] \models \varphi$ for some path λ starting from q;

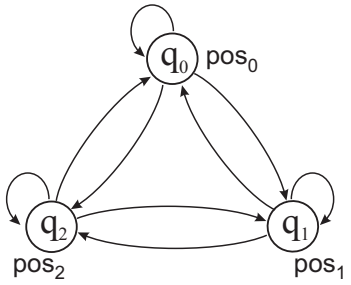

Fig. 2. Two robots and a carriage: transition system M_2 that models possible movements of the carriage.

$M, q \models \mathsf{E}\mathrm{G}\varphi$ iff there is a path λ starting from q such that $(\forall i \geq 0)\ M, \lambda[i] \models \varphi$;
$M, q \models \mathsf{E}\varphi\mathrm{U}\psi$ iff there is a path λ starting from q such that $(\exists j \geq 0)\big(M, \lambda[j] \models \psi$ and $(\forall 0 \leq i < j)\ M, \lambda[i] \models \varphi\big)$.

Example 3. A possible transition system for the robots scenario is depicted in Figure 2. In that model, we have for instance $M_2, q_0 \models \mathsf{EF}\,\mathsf{pos}_1$: in state q_0, there is a path such that the carriage will reach position 1 sometime in the future. The same is clearly not true for *all* paths, so we also have that $M_2, q_0 \models \neg\mathsf{AF}\,\mathsf{pos}_1$.

Temporal and dynamic dimensions have been combined with other modalities, e.g. in the well-known *BDI logics* of beliefs, desires, and intentions [16, 84]. We will present simple extensions of LTL and CTL with epistemic modalities in Section 2.5, and discuss verification of temporal-epistemic logic in later sections.

Modal μ-calculus. Propositional modal μ-calculus was introduced by D. Kozen [62]. Let $\mathcal{PV}$ be a set of propositional variables and FV be a set of fixed-point variables. The language of modal μ-calculus L_μ is defined by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{AX}\varphi \mid \mathsf{EX}\varphi \mid Z \mid \mu Z.\varphi(Z) \mid \nu Z.\varphi(Z),$$

where p ranges over $\mathcal{PV}$, Z – over FV, and $\varphi(Z)$ is a modal μ-calculus formula syntactically monotone in the fixed-point variable Z, i.e., all the free occurrences of Z in $\varphi(Z)$ fall under an even number of negations.

Let $M = (St, \to, V)$ be a Kripke model. Notice that the set 2^{St} of all subsets of St forms a lattice under the set inclusion ordering. Each element St' of the lattice can also be thought of as a *predicate* on St, where this predicate is viewed as being true for exactly the states in St'. The least element in the lattice is the empty set, which we also refer to as **false**, and the greatest element in the lattice is the set St, which we sometimes write as **true**. A function ζ mapping 2^{St} to 2^{St} is called a *predicate transformer*. A set $St' \subseteq St$ is a *fixed point* of a function $\zeta : 2^{St} \longrightarrow 2^{St}$ if

$$\zeta(St') = St'.$$

Whenever ζ is *monotonic*, i.e., $S_1 \subseteq S_2$ implies $\zeta(S_1) \subseteq \zeta(S_2)$, where $S_1, S_2 \subseteq St$, it has the least fixed point denoted $\mu Z.\zeta(Z)$ and the greatest fixed point-denoted $\nu Z.\zeta(Z)$. When $\zeta(Z)$ is also $\bigcup$-continuous, i.e., $S_1 \subseteq S_2 \subseteq \ldots$ implies $\zeta(\bigcup_{i\geq 0} S_i) = \bigcup_{i\geq 0} \zeta(S_i)$, then

$$\mu Z.\zeta(Z) = \bigcup_{i\geq 0} \zeta^i(\mathbf{false}).$$

When $\zeta(Z)$ is also $\bigcap$-continuous, i.e., $S_1 \supseteq S_2 \supseteq \ldots$ implies $\zeta(\bigcap_{i\geq 0} S_i) = \bigcap_{i\geq 0} \zeta(S_i)$, then

$$\nu Z.\zeta(Z) = \bigcap_{i\geq 0} \zeta^i(\mathbf{true})$$

(see [94]).

The semantics of L_μ is given inductively for each formula φ of L_μ, a model M, a valuation $\mathcal{E} : FV \longrightarrow 2^S$ of the fixed-point variables (called an *environment*), and a state $q \in St$:

$$\begin{array}{lll}
M,\mathcal{E},q \models p & \text{iff} & q \in V(p), \text{ for } p \in \mathcal{PV},\\
M,\mathcal{E},q \models \neg\varphi & \text{iff} & M,\mathcal{E},q \not\models \varphi,\\
M,\mathcal{E},q \models \varphi \wedge \psi & \text{iff} & M,\mathcal{E},q \models \varphi \text{ and } M,\mathcal{E},q \models \psi,\\
M,\mathcal{E},q \models \mathrm{AX}\varphi & \text{iff} & (\forall q' \in St)((q{\rightarrow}q') \Rightarrow (M,\mathcal{E},q' \models \varphi)),\\
M,\mathcal{E},q \models \mathrm{EX}\varphi & \text{iff} & (\exists q' \in St)((q{\rightarrow}q') \wedge M,\mathcal{E},q' \models \varphi),\\
M,\mathcal{E},q \models Z & \text{iff} & q \in \mathcal{E}(Z), \text{ for } Z \in FV,\\
M,\mathcal{E},q \models \mu Z.\varphi(Z) & \text{iff} & q \in \bigcap\{U \subseteq St \mid \\
 & & \quad \{q' \in St \mid M,\mathcal{E}[Z \leftarrow U],q' \models \varphi\} \subseteq U\},\\
M,\mathcal{E},q \models \nu Z.\varphi(Z) & \text{iff} & q \in \bigcup\{U \subseteq St \mid \\
 & & \quad U \subseteq \{q' \in St \mid M,\mathcal{E}[Z \leftarrow U],q' \models \varphi\}\},
\end{array}$$

where $\varphi, \psi \in L_\mu$, and $\mathcal{E}[Z \leftarrow U]$ is like $\mathcal{E}$ except that it maps Z to U. Now, we define $M, q \models \varphi$ iff $M,\mathcal{E},q \models \varphi$ for each environment $\mathcal{E}$.

It is known that both CTL and CTL* can be translated into modal μ-calculus [62]. For example, we give characterisations of basic CTL modalities in terms of modal μ-calculus formulas:

- $\mathrm{A}(\varphi \mathrm{U} \psi) \equiv \mu Z.(\psi \vee (\varphi \wedge \mathrm{AX}Z))$,
- $\mathrm{E}(\varphi \mathrm{U} \psi) \equiv \mu Z.(\psi \vee (\varphi \wedge \mathrm{EX}Z))$,
- $\mathrm{AG}\varphi \equiv \nu Z.(\varphi \wedge \mathrm{AX}Z)$,
- $\mathrm{EG}\varphi \equiv \nu Z.(\varphi \wedge \mathrm{EX}Z)$.

The translation of CTL* to modal μ-calculus is more involved and can be found in [24]. It is worth noticing that the translations are important in practice because correctness specifications written in logics such as CTL or CTL* are often much more readable than specifications written directly in modal μ-calculus.

References: Dynamic logic is treated extensively in [37]; readers interested in temporal logic are referred to [23, 29]. An introduction to μ-calculus can be found in [93].

2.5 Combining Time and Knowledge

In this section we discuss how the interaction between knowledge and time can be captured in modal logic. To this end, we combine epistemic and temporal dimensions in a multi-modal logic. The main proposals that we are going to consider is LTLK and CTLK, which are straightforward combinations of respectively LTLK, CTL, and standard epistemic logic.

LTLK. The language of LTLK includes all the operators of LTL and epistemic logic. Moreover, its semantics is based on Kripke models which include both temporal and epistemic accessibility relations: $M = \langle St, R, \sim_1, ..., \sim_k, V\rangle$. The semantics of LTLK simply combines the clauses for LTL and modal epistemic logic:

$$\begin{aligned}
\lambda &\models p \text{ iff } \lambda[0] \in V(p);\\
\lambda &\models \neg\varphi \text{ iff } \lambda \not\models \varphi,\\
\lambda &\models \varphi \wedge \psi \text{ iff } \lambda \models \varphi \text{ and } \lambda \models \psi;\\
\lambda &\models \mathrm{X}\varphi \text{ iff } \lambda_1 \models \varphi;\\
\lambda &\models \varphi \mathrm{U} \psi \text{ iff } (\exists j \geq 0)\big(\lambda_j \models \psi \text{ and } (\forall 0 \leq i < j)\ \lambda_i \models \varphi\big);\\
\lambda &\models \mathrm{K}_i\varphi \text{ iff } \lambda'_j \models \varphi \text{ for all paths } \lambda' \text{ and all } (j \geq 0) \text{ s.t. } \lambda[0] \sim_i \lambda'[j],
\end{aligned}$$

where $1 \leq i \leq k$ and $M, q \models \varphi$ iff $M, \lambda_0 \models \varphi$ for all the paths λ starting at q.

CTLK. The language of CTLK includes all the operators of CTL and epistemic logic. Moreover, its semantics is based on Kripke models which include both temporal and epistemic accessibility relations: $M = \langle St, R, \sim_1, ..., \sim_k, V\rangle$. Like for LTLK, the semantics of CTLK takes the union of both respective semantics:

$M, q \models p$ iff $q \in V(p)$;
$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;
$M, q \models \mathsf{EX}\varphi$ iff $M, \lambda[1] \models \varphi$ for some path λ starting from q;
$M, q \models \mathsf{EG}\varphi$ iff there is a path λ starting from q such that $(\forall i \geq 0)\ M, \lambda[i] \models \varphi$;
$M, q \models \mathsf{E}\varphi\mathrm{U}\psi$ iff there is a path λ starting from q such that $(\exists j \geq 0)\big(M, \lambda[j] \models \psi$ and $(\forall 0 \leq i < j)\ M, \lambda[i] \models \varphi\big)$.
$M, q \models \mathrm{K}_i\varphi$ iff $M, q \models \varphi$ for every q' such that $q \sim_i q'$, for $1 \leq i \leq k$.

Example 4. Let M_3 be the temporal-epistemic model of robots and carriage that combines states and relations from models M_1, M_2 (Figures 1 and 2). Now, $M_3, q_0 \models \mathrm{K}_1\mathsf{EF}\,\mathsf{pos}_2$: robot 1 knows that the carriage can get to position 2 eventually. Moreover, $M_3, q_0 \models \mathsf{EX}\bigvee_{i=0,1,2}(\mathsf{pos}_i \rightarrow \mathrm{K}_1\mathsf{pos}_i)$: it is possible that in the next moment robot 1 will know its position precisely.

Interpreted systems. A well known study of the interplay between knowledge and time has been presented in [28]. The proposal builds on a notion of *global states*, defined formally as follows. Firstly, each agent $i \in \mathbb{A}\mathrm{gt} = \{1, ..., k\}$ has a set of *local states* St_i. Every *global state* is represented by a tuple of local

states $\langle q_1, ..., q_k \rangle$ corresponding to all agents. Thus, the global state space St is (a subset of) the product $St_1 \times ... \times St_k$. It has been argued in many places that interpreted systems provide more "grounded" semantics for agents' knowledge than abstract Kripke models. This is because starting from the local state spaces makes it clearer how the epistemic model should be actually constructed.

It is usually assumed in interpreted systems that each agent has access only to its own local state, i.e.:

$$\langle q_1, ..., q_k \rangle \sim_i \langle q'_1, ..., q'_k \rangle \text{ iff } q_i = q'_i.$$

The temporal dimension is added by considering *runs*, i.e., sequences of global states:

$$r : \mathbb{N} \to St_1 \times ... \times St_k.$$

A *system* is a set of such runs. An *interpreted system* is a system plus a valuation of propositions: $\langle \mathcal{R}, V \rangle$. Given an interpreted system $\mathcal{I} = \langle \mathcal{R}, V \rangle$, a *point* in $\mathcal{I}$ is a pair $\langle r, m \rangle$ where r is a run and $m \in \mathbb{N}$. Epistemic equivalence between points is defined as follows:

$$\langle r, m \rangle \sim_i \langle r', m' \rangle \text{ iff } r(m) \sim_i r'(m').$$

Now, all epistemic modalities can be interpreted as before, e.g.:

$$\mathcal{I}, r, m \models \mathrm{K}_i\varphi \text{ iff } \mathcal{I}, r', m' \models \varphi \text{ for all } \langle r', m' \rangle \text{ such that } \langle r, m \rangle \sim_i \langle r', m' \rangle.$$

Moreover, the standard temporal operators of LTL can be interpreted, too:

- $\mathcal{I}, r, m \models \mathrm{X}\varphi$ iff $\mathcal{I}, r, m + 1 \models \varphi$,
- $\mathcal{I}, r, m \models \varphi \mathrm{U} \psi$ iff $\mathcal{I}, r, m' \models \psi$ for some $m' > m$ and $\mathcal{I}, r, m'' \models \varphi$ for all m'' such that $m \leq m'' < m'$.

Finally, the semantics of path quantifiers from CTL* can be defined in interpreted systems as follows:

- $\mathcal{I}, r, m \models \mathsf{E}\varphi$ iff there is r' such that $r'[0..m] = r[0..m]$ and $\mathcal{I}, r', m \models \varphi$.

It should be pointed out that the semantics of knowledge presented above is only one of several possibilities. It encodes the assumption that agents are *memoryless* in the sense that they do not have "external" memory of past events; the whole memory of an agent is encapsulated in its local state. The other extreme is to assume that local states represent the agents' observations rather than knowledge, and that agents have *perfect recall* of everything that has been observed. If we additionally assume the existence of a global universally accessible clock, the semantics of knowledge can be defined as follows:

$$\langle r, m \rangle \approx_i \langle r', m' \rangle \text{ iff } m = m' \text{ and } r(j) \sim_i r'(j) \text{ for all } j \leq m,$$
$$\mathcal{I}, r, m \models \mathrm{K}_i\varphi \text{ iff } \mathcal{I}, r', m' \models \varphi \text{ for all } \langle r', m' \rangle \text{ such that } \langle r, m \rangle \approx_i \langle r', m' \rangle.$$

Interpreted systems have been applied to modeling of *distributed systems, knowledge bases, message passing systems*, and more specifically to phenomena like *perfect recall, synchrony and asynchrony* etc.

References: Possible interactions between the temporal and epistemic dimensions are studied extensively in [28].

3 Introduction to Model Checking for Knowledge and Time

In this section, we look at standard non-symbolic model checking algorithms for temporal and temporal-epistemic logics. We start with some notes on verification and complexity classes in Section 3.1. Basic algorithms and complexity results for verification of temporal logics are presented in Section 3.2. Then, we discuss the model checking algorithm exploiting the fixed-point characterization of CTLK in Section 3.3.

3.1 Complexity of Verificatin

Within the materials we consider both practical algorithms for verifying properties of MAS, and the theoretical complexity of these problems. Here is a short (and rather informal) description of the most relevant complexity classes. A formal introduction can be found in any textbook on complexity theory (cf. e.g. [72]).

- **P**: problems solvable in *polynomial time* by a *deterministic* Turing machine,
- **NP**: problems solvable in *polynomial time* by a *nondeterministic* Turing machine,
- $\mathbf{\Sigma_n^P}$ / $\mathbf{\Pi_n^P}$ / $\mathbf{\Delta_n^P}$: problems solvable in polynomial time with use of adaptive queries to an *n-level oracle*,
- **PSPACE**: problems solvable by queries to a multilevel oracle with unbounded "height",
- **EXPTIME**: problems solvable in *exponential time* by a deterministic Turing machine.

Of course, theoretical complexity has many deficiencies: it refers only to the worst (hardest) instance in the set, neglects coefficients in the function characterizing the complexity, etc. However, it often gives a good indication of the inherent hardness of the problem in terms of *scalability*. For low complexity classes, scaling up from small instances of the problem to larger instances is relatively easy. For high complexity classes, this is not the case anymore.

The process of verification (so called *model checking*) answers whether a given formula φ is satisfied in a state q of model M. Formally, *local model checking* is the decision problem that determines membership in the set

$$\mathsf{MC}(\mathcal{L}, \mathsf{Struc}, \models) = \{(M, q, \varphi) \in \mathsf{Struc} \times St \times \mathcal{L} \ \mid \ M, q \models \varphi\},$$

where $\mathcal{L}$ is a logical language, Struc is a class of (pointed) models for $\mathcal{L}$, and $\models$ is a semantic satisfaction relation compatible with $\mathcal{L}$ and Struc.[1]

It is often useful to compute the set of states in M that satisfy formula φ instead of checking if φ holds in a particular state. This variant of the problem is known as *global model checking*. It is easy to see that, for the settings we

[1] We omit parameters if they are clear from the context.

consider here, the complexities of local and global model checking coincide, and the algorithms for one variant of model checking can be adapted to the other variant in a simple way. As a consequence, we will use both notions of model checking interchangeably.[2]

References: For a comprehensive textbook on model checking, see e.g. [15].

3.2 Verifying Temporal Logic

An excellent survey on the model checking complexity of temporal logics has been presented in [86]. Here, we only recall the most relevant results before turning our focus to actual algorithms in Section 3.

Let M be a Kripke model and q be a state in the model. Model checking a CTL formula φ in M, q determines whether $M, q \models \varphi$, i.e., whether φ holds in M, q. The same applies to model checking CTL*. For LTL, checking $M, q \models \varphi$ means that we check the *validity* of φ in the pointed model M, q, i.e., whether φ holds *on all the paths* in M that start from q (equivalent to CTL* model checking of formula $\mathsf{A}\varphi$ in M, q.

It has been known for a long time that the formulae of CTL can be model-checked in time linear with respect to the size of the model and the length of the formula [14], whereas formulae of LTL and CTL* are significantly harder to verify. The size of the model M, denoted by $|M|$, is defined by the sum of the number of its states and its transitions $|St| + |R|$.

Theorem 1 (CTL [14, 86]). *Model checking CTL is* **P***-complete, and can be done in time* $\mathbf{O}(|M| \cdot |\varphi|)$.

Proof (Sketch). The algorithm (for an extension of CTL) determining the states in a model at which a given formula holds is presented in Section 3.3. The lower bound (**P**-hardness) can be for instance proven by a reduction of the tiling problem [86]. □

Theorem 2 (LTL [92, 67, 102]). *Model checking LTL is* **PSPACE***-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

Proof (Sketch). We sketch here the approach of [102]. Firstly, given an LTL-formula φ, a Büchi automaton $\mathcal{A}_{\neg\varphi}$ of size $2^{\mathbf{O}(|\varphi|)}$ accepting exactly the runs (infinite paths of states) satisfying $\neg\varphi$ is constructed. The pointed Kripke model M, q can directly be interpreted as a Büchi automaton $\mathcal{A}_{M,q}$ of size $\mathbf{O}(|M|)$ accepting all possible runs in the Kripke model starting in q. Then, the model checking problem reduces to the non-emptiness check of $L(\mathcal{A}_{M,q}) \cap L(\mathcal{A}_{\neg\varphi})$, which can be done in time $\mathbf{O}(|M|) \cdot 2^{\mathbf{O}(|\varphi|)}$ by constructing the product automaton. Notice that the non-emptiness can be checked in linear time w.r.t. to the size of the automaton. A **PSPACE**-hardness proof can be for instance found in [92]. □

[2] The only logic mentioned in this chapter, for which the complexities of global and model checking differ, is Constructive Strategic Logic from Section 4.4. However, we do not discuss the model checking problem for CSL in the materials.

MC	m, l
Epistemic logic	**P**-complete
PDL	**P**-complete
CTL	**P**-complete
LTL	**PSPACE**-complete
CTL*	**PSPACE**-complete

SAT	l
Epistemic logic	**PSPACE**-complete
PDL	**PSPACE**-complete
CTL	**EXPTIME**-complete
LTL	**PSPACE**-complete
CTL*	**2EXPTIME**-complete

Fig. 3. Basic complexity results: model checking (left) and satisfiability (right)

The hardness of CTL* model checking is immediate from Theorem 2 as LTL can be seen as a fragment of CTL*. For the proof of the upper bound one combines the CTL and LTL model checking techniques. Consider a CTL* formula φ which contains a state subformula $\mathsf{E}\psi$, where ψ is a pure LTL formula. Firstly, we can use the LTL model checking to determine all the states which satisfy $\mathsf{E}\psi$ (these are all states q in which the LTL formula $\neg\psi$ is *not* true) and label them by a fresh propositional symbol, say p, and replace $\mathsf{E}\psi$ in φ by p as well. Applying this procedure recursively yields a pure CTL formula, which can be verified in polynomial time. Hence, the procedure can be implemented by an oracle machine of type $\mathbf{P}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$ (the LTL model checking algorithm might be employed polynomially many times). Thus, the complexity for CTL* is the same as for LTL.

Theorem 3 (CTL* [14, 25]). *Model checking CTL* is* **PSPACE***-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

Figure 3 presents the basic complexity results for model checking of temporal and epistemic logics. For comparison, we also list complexities of analogous satisfiability problems. The input of the SAT problem is given as a formula of the logic, and its size is measured in the length of the formula. The input of the model checking problem is given as the formula and the model; the size of an input instance is measured as $m \cdot l$, where m is the number of transitions in the model, and l is the length of the formula.

References: Readers interested in basic complexity results for model checking temporal logics are referred to the excellent survey [86].

3.3 Fixed-point Verification for CTL and CTLK

Symbolic and non-symbolic model checking methods can exploit the fixed-point characterization of CTLK formulas. These methods operate on sets of states contrary to the state labeling algorithm operating on single states of the model. In what follows by $\overline{\mathrm{C}}_\mathrm{A}$ we denote the modality dual to C_A, i.e., $\overline{\mathrm{C}}_\mathrm{A}\varphi \stackrel{def}{=} \neg\mathrm{C}_\mathrm{A}\neg\varphi$. Similarly, for $\overline{\mathrm{E}}_\mathrm{A}$ and $\overline{\mathrm{D}}_\mathrm{A}$. We do not discuss here algorithms for the operators $\overline{\mathrm{E}}_\mathrm{A}$ and $\overline{\mathrm{D}}_\mathrm{A}$ as these are straightforward given an algorithm for $\overline{\mathrm{K}}_i$. Then, labelling of the states with the subformulas or computation of OBDD representation of

a CTLK formula uses the standard algorithms for computing the minimal and the maximal fixpoints as follows.

- $\mathrm{EG}\varphi \equiv \varphi \wedge \mathrm{EXEG}\varphi$,
- $\underline{\mathrm{E}}(\varphi \mathrm{U} \psi) \equiv \psi \vee (\varphi \wedge \mathrm{E}(\varphi \mathrm{U} \psi))$,
- $\overline{\mathrm{C}}_{\mathrm{A}}\varphi \equiv \varphi \vee \overline{\mathrm{E}}_{\mathrm{A}}\overline{\mathrm{C}}_{\mathrm{A}}\varphi$.

Let $[\![\varphi]\!] = \{q \in St \mid q \models \varphi\}$. Then, we have:

- $[\![EG\varphi]\!] = [\![\varphi]\!] \cap [\![\mathrm{EXEG}\varphi]\!]$,
- $[\![\underline{\mathrm{E}}(\varphi \mathrm{U} \psi)]\!] = [\![\psi]\!] \cup ([\![\varphi]\!] \cap [\![\mathrm{EXE}(\varphi \mathrm{U} \psi)]\!])$
- $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!] = [\![\varphi]\!] \cup [\![\overline{\mathrm{E}}_{\mathrm{A}}\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!])$

For each subset $X \subseteq St$, we can easily define algorithms computing the following sets:

- $pre(X) = \{q \in St \mid (\exists q' \in X)\ q \to q'\}$,
- $indis_i(X) = \{q \in St \mid (\exists q' \in X)\ q \sim_i q'\}$, and
- $indis^E_{\mathrm{A}}(X) = \{q \in St \mid (\exists q' \in X)\ q \sim^E_{\mathrm{A}} q'\}$.

Then, we have:

- $[\![EG\varphi]\!] = [\![\varphi]\!] \cap pre([\![\mathrm{EG}\varphi]\!])$,
- $[\![\underline{\mathrm{E}}(\varphi \mathrm{U} \psi)]\!] = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre([\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!]))$,
- $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!] = [\![\varphi]\!] \cup indis^E_{\mathrm{A}}([\![\overline{\mathrm{C}}_A\varphi]\!])$.

Next, define three functions on 2^{St}, which fixed points are equal to respectively $[\![\mathrm{EG}\varphi]\!]$, $[\![\mathrm{E}(\varphi \mathrm{U} \psi)]\!]$, and $[\![\overline{\mathrm{C}}_{\mathrm{A}}\varphi]\!]$.

1. $\tau_{\mathrm{EG}\varphi}(X) = [\![\varphi]\!] \cap pre(X)$,
2. $\tau_{\mathrm{E}(\varphi \mathrm{U} \psi)}(X) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre(X))$,
3. $\tau_{\overline{\mathrm{C}}_A\varphi}(X) = [\![\varphi]\!] \cup indis^E_{\mathrm{A}}(X)$.

Since $\mathrm{EG}\varphi$ is the maximal fixpoint of $\tau_{\mathrm{EG}\varphi}(X)$, it can be computed as $\tau^k_{\mathrm{EG}\varphi}(St)$ for some finite k. Since $\mathrm{E}(\varphi \mathrm{U} \psi)$ is the minimal fixpoint of $\tau_{\mathrm{E}(\varphi \mathrm{U} \psi)}(X)$ it can be computed as $\tau^l_{\mathrm{E}(\varphi \mathrm{U} \psi)}(\emptyset)$ for some finite l. Similarly, for $\tau_{\overline{\mathrm{C}}_A\varphi}(X)$. The above characterization can be now used for defining a model checking algorithm mchk for the formulas of CTLK.

$mchk(M, \varphi)$ {
if $\varphi \in PV$, then return $V^{-1}(\varphi)$,
if $\varphi = \neg\psi$, then return $St \setminus mchk(M, \psi)$,
if $\varphi = \varphi_1 \wedge \varphi_2$, then return $mchk(M, \varphi_1) \cap mchk(M, \varphi_2)$,
if $\varphi = \mathrm{EX}\psi$, then return $mchk_{\mathrm{EX}}(M, \psi)$,
if $\varphi = \overline{\mathrm{K}}_i\psi$, then return $mchk_{\overline{\mathrm{K}}_i}(M, \psi)$,
if $\varphi = \mathrm{EG}\psi$, then return $mchk_{\mathrm{EG}}(M, \psi)$,
if $\varphi = \underline{\mathrm{E}}(\phi \mathrm{U} \psi)$, then return $mchk_{\mathrm{EU}}(M, \phi, \psi)$,
if $\varphi = \overline{\mathrm{C}}\psi$, then return $mchk_{\overline{\mathrm{C}}}(M, \psi)$.
}

$mchk_{\mathrm{EX}}(M, \psi)\{$
$X := mchk(M, \psi);$
$Y := pre(X);$
return Y $\};$

$mchk_{\mathrm{EG}}(M, \psi)\{$
$X := mchk(M, \psi);$
$Y := St;$
$Z := \emptyset;$
while $(Z \neq Y)\{$
$Z := Y;$
$Y := X \cap pre(Y)\}$
return Y $\};$

$mchk_{\overline{\mathrm{C}}}(M, \psi)\{$
$Y := mchk(M, \psi);$
$Z := \emptyset;$
$W := St;$
while $(Z \neq W)\{$
$W := Z;$
$Z := Y \cup indis^{E}_{\mathrm{A}}(Z))\}$
return Z $\};$

$mchk_{\mathrm{EK}_i}(M, \psi)\{$
$X := mchk(M, \psi);$
$Y := indis_i(X);$
return Y $\};$

$mchk_{\mathrm{EU}}(M, \psi_1, \psi_2)\{$
$X := mchk(M, \psi_1);$
$Y := mchk(M, \psi_2);$
$Z := \emptyset;$
$W := St;$
while $(Z \neq W)\{$
$W := Z;$
$Z := Y \cup (X \cap pre(Z))\}$
return Z $\};$

References: The original state labelling algorithm for CTL was introduced in [13]. More information on model checking CTL can be found in [76, 43]. CTLK is treated extensively in [68].

For other non-symbolic approaches to model checking temporal-epistemic properties, one may e.g. refer to [41]. There, a translation of a fragment of LTLK to LTL is proposed, and the SPIN model checker is used for verification.

4 Specification of Agents and Their Teams

In this section we present modal logics that can be used to reason about strategies and abilities of agents in game-like scenarios. We begin with a short exposition of the game-theoretic inspiration (normal form models of games). However, it should be pointed out that the current modal logic-based approaches to reasoning about strategic play are very weak in game-theoretic sense. They are based on worst case analysis ("surely winning") and binary winning conditions, and hence roughly correspond to maxmin analysis in two-player zero-sum games with binary payoffs. Some attempts have been made at incorporating more sophisticated solution concepts like Nash equilibrium, dominance, Paret-optimality etc. [39, 38, 95, 103, 12] as well as probabilistic features like chance nodes and mixed strategies [19, 46, 10, 87, 42]. Still, none of them matches the elegance and simplicity of the way models and solution concepts are defined in game theory.

4.1 Games and Strategies

Interactions between autonomous and rational agents acting strategically have been extensively studied in the field of *game theory*. The models used in game theory can be categorised into two types: *non-cooperative* games, in which the possible actions of individual players are taken as primitives, and *coalitional* (or *cooperative*) games, in which the possible joint actions of groups of players are taken as primitives.

A standard model in non-cooperative game theory is that of a *strategic game*. In a strategic game, it is assumed that each agent chooses her future actions (her strategy) once and for all at the beginning of the game, and that all agents do this simultaneously. Formally, a strategic game is a tuple $G = (\mathbb{A}\mathrm{gt}, \{\Sigma_i : i \in \mathbb{A}\mathrm{gt}\}, o, St, \{\succeq_i : i \in \mathbb{A}\mathrm{gt}\})$, where $\mathbb{A}\mathrm{gt}$ is the set of agents, St is a set of states, Σ_i is the set of actions (or strategies) available to agent i, o associates an outcome state $o(a_1, \ldots, a_n) \in St$ with every tuple of actions $(a_1, \ldots, a_n) \in \times_{i \in \mathbb{A}\mathrm{gt}} \Sigma_i$, and $\succeq_i \subseteq St \times St$ is a preference relation (complete, reflexive, transitive) over the outcome states for agent i. Leaving out the preference relations, we get a *strategic game frame* $G = (\mathbb{A}\mathrm{gt}, \{\Sigma_i : i \in \mathbb{A}\mathrm{gt}\}, o, St)$.

Cooperation in games. Henceforth, a *coalition* is simply a group of agents. Coalitional actions are tuples of individual actions, one per member of the coalition; formally, $\Sigma_C = \prod_{i \in C} \Sigma_i$. An *effectivity function* models the power that agents can obtain by forming coalitions. Given a set of states X, we can say that a coalition C is *effective* for X if C can cooperate to ensure that the outcome will be in X. Formally, an effectivity function for a set of players $\mathbb{A}\mathrm{gt}$ over a set of states St is a function

$$E : 2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{St}}$$

giving the sets of outcomes $E(C)$ for which each coalition C is effective. Game frame G induces an associated effectivity function E_G as follows:

$$X \in E_G(C) \Leftrightarrow \exists \sigma_C \in \Sigma_C \; \forall \sigma_{\mathbb{A}\mathrm{gt} \setminus C} \in \Sigma_{\mathbb{A}\mathrm{gt} \setminus C} \; . \; o(\sigma_C, \sigma_{\mathbb{A}\mathrm{gt} \setminus C}) \in X.$$

Effectivity functions induced from game frames in this way are called α-*effectivity functions* in social choice theory.

References: [71] is a standard textbook in game theory. [66] approaches game-theoretical concepts from multi-agent systems perspective. Effectivity functions and related correspondence results are studied in [73, 33]

4.2 Coalition Logic

Modal logics of strategic ability form one of the fields where logic and game theory can successfully meet. Marc Pauly's *Coalition Logic* (CL) formalises reasoning about the abilities of coalitions. The language of CL extends propositional logic with a modality $[C]$ for each coalition C. The intended meaning of $[C]\varphi$ is that C can make the outcome of the game satisfy φ.

Formally, the language is interpreted over *coalition models*

$$M = (St, E, V)$$

where St is a set of states, $V : \mathcal{PV} \to 2^{St}$ an interpretation of atomic propositions $\mathcal{PV}$ in the states, and

$$E : St \to (2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{St}})$$

assigns to each $q \in St$ an effectivity function $E(q)$.[3] The semantics of $[C]\varphi$ is defined as follows:

$$M, q \models [C]\phi \text{ iff } \phi^M \in E(q)(C)$$

where $\phi^M = \{q \in St : M, q \models \phi\}$.

Since the coalition models being used are usually based on α-effectivity functions of some strategic game, formulae of Coalition Logic can be alternatively seen as statements about strategic game forms.

References: Coalition Logic was introduced and studied in [73].

4.3 ATL

Coalition Logic allows only to reason about agents' outcomes in strategic (one-step) games. What if a game consists of multiple steps, like in the case of extensive form games? In this section, we focus on *Alternating-time Temporal Logic (ATL)*, one of the most important logics of time and strategies that have emerged in recent years. ATL can be seen as a generalization of the branching time temporal logic CTL (Computation Tree Logic), in which path quantifiers (E: "there is a path", A: "for every path") are replaced with *strategic quantifiers* $\langle\langle A\rangle\rangle$. The formula $\langle\langle A\rangle\rangle\varphi$, where A is a coalition of agents, expresses the claim that A has a collective strategy to enforce the temporal property φ throughout the interaction. Moreover, ATL formulas include the usual temporal operators: X ("in the next state"), G ("always from now on"), F ("sometime in the future"), and U ("until").

[3] $E(q)$ is usually required to satisfy some structural constraints, cf. [73] for details.

Several semantics have been introduced for ATL, of which the one based on *concurrent game structures* (CGS) is perhaps the most popular. A concurrent game structure $M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, o, V\rangle$ includes: a nonempty finite set of agents $\mathbb{A}\mathrm{gt} = \{1, ..., k\}$; a nonempty set of global states St of the system; a set of (atomic) actions Act; function $d : \mathbb{A}\mathrm{gt} \times St \to 2^{Act}$ defining the set of actions available to an agent in a state; a valuation of propositions $V : St \to 2^{\mathcal{PV}}$; finally, a (deterministic) transition function o which assigns an outcome state $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to state q and a tuple of actions $\langle \alpha_1, \ldots, \alpha_k\rangle$ that can be executed by $\mathbb{A}\mathrm{gt}$ in q. Note that a concurrent game structure can be seen as a collection of interconnected strategic game forms, or, alternatively, as a generalized extensive game form (plus a valuation of propositions).

A *strategy* of agent a is a conditional plan that specifies what a is going to do for every 'possible situation'. For an agent with no implicit recall of the past, a strategy can be thus represented by function $s_a : St \to Act$ such that $s_a(q) \in d_a(q)$, i.e., a function that specifies a valid choice of action for each *state* of the system. For agents with perfect recall, this would be $s_a : St^+ \to Act$ such that $s_a(q_0q_1...q_i) \in d_a(q_i)$, i.e., one that specifies a valid choice of action for each *history* of the game. A *collective strategy* s_A for a group of agents A is a tuple of strategies, one per agent from A. Then, by $out_M(q, s_A)$ (or just $out(q, s_A)$) we denote the set of all paths that may result from agents A executing strategy s_A from state q onward. The semantics of strategic quantifiers is defined as follows:

$M, q \models \langle\langle A\rangle\rangle\gamma$ iff there is a strategy s_A, such that $M, \lambda \models \gamma$ for every $\lambda \in out(q, s_A)$;
$M, \lambda \models \mathrm{X}\gamma$ iff $M, \lambda_1 \models \gamma$;
$M, \lambda \models \mathrm{G}\gamma$ iff $M, \lambda_i \models \gamma$ for all $i \geq 0$;
$M, \lambda \models \gamma_1\mathrm{U}\gamma_2$ iff for some $i \geq 0$ ($\lambda_i \models \gamma_2$ and for all $(0 \leq j < i)$ $\lambda_j \models \gamma_1$).

Additionally, the "sometime" modality can be defined as: $\mathrm{F}\gamma \equiv \top\mathrm{U}\gamma$.

Note that concurrent game structures do not allow to represent any kind of uncertainty, and strategies can freely assign arbitrary choices in states (resp. histories). In consequence, ATL can be seen as a logic for reasoning about agents with *perfect information*. That is, it is implicitly assumed that each agent always knows the current state of the game precisely, and in particular he can distinguish between any pair of global states. The notions of perfect vs. imperfect information will be address more formally in Section 4.4.

The above clauses define the semantics of the full version of alternating-time logic, usually called ATL* for historical reasons. It must be noted, however, that the typical variant of ATL, used in the literature, is restricted to formulae in which every temporal operator is immediately preceded by exactly one cooperation modality. We will usually refer to the restricted variant as "vanilla" ATL.

Example 5 (Robots and Carriage, ctd.). Models and languages in Section 2 enabled studying evolution of the robots and carriage as a whole. However, they did not allow us to represent *who* can achieve *what*, and how possible actions of the agents interact. Concurrent game structure M_4, presented in Figure 4, fills the gap. We assume that each robot can either push (action *push*) or refrain from

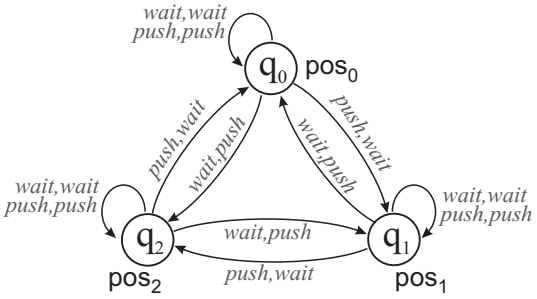

Fig. 4. The robots and the carriage: a concurrent game structure M_4.

pushing (action *wait*). Moreover, both robots use the same force when pushing. Thus, if they push simultaneously or wait simultaneously, the carriage does not move. When only one of the robots is pushing, the carriage moves accordingly.

As the outcome of each robot's action depends on the current action of the other robot, no agent can make sure that the carriage moves to any particular position. So, we have for example that $M_4, q_0 \models \neg\langle\langle 1\rangle\rangle \mathrm{F}\,\mathsf{pos}_1$. On the other hand, the agent can at least make sure that the carriage will *avoid* particular positions. For instance, it holds that $M_4, q_0 \models \langle\langle 1\rangle\rangle \mathrm{G}\neg\mathsf{pos}_1$, the right strategy being $s_1(q_0) = wait, s_1(q_2) = push$ (the action in q_1 is irrelevant).

We observe that ATL syntactically embeds two important logics. First, the branching time logic CTL can be seen as a strategic logic with a very limited set of strategic quantifiers (as the CTL path quantifiers can be embedded in ATL with the following definitions: $\mathsf{E}\varphi \equiv \langle\langle \mathbb{A}\mathrm{gt}\rangle\rangle\varphi$, $\mathsf{A}\varphi \equiv \langle\langle \varnothing\rangle\rangle\varphi$). Second, Coalition Logic can be seen as the "next"-fragment of ATL, with the following embedding of the central modality of CL: $[A]\varphi \equiv \langle\langle A\rangle\rangle \mathrm{X}\varphi$. We also point out that ATL embeds CTL semantically because CTL can be seen as the single-agent fragment of ATL. More precisely, CTL is equivalent to ATL interpreted over structures that include only one player ("the system").

Bisimulation in ATL. The concept of *bisimulation* is central in modal logic, and is for example very useful in order to study the key meta-logical property of *expressiveness*. Bisimulation is a relationship between semantic structures, and it is typically the case that bisimilar structures satisfy exactly the same formulae. Then, we immediately know something about the expressiveness of the logical language: if two structures are bisimilar and one of them has some property while the other has not, then that property is not expressible in the logical language. The following is an adaption [2] of the standard modal logic notion of a bisimulation to CGSs.

First, let $D(q, C) = \prod_{i\in C} d_i(q)$ denote the set of *joint actions* of coalition C in state q. For $\boldsymbol{a}_C \in D(q, C)$, let

$$next_M(q, \boldsymbol{a}_C) = \{o(q, \alpha) : \text{ there is } \alpha \in D(q, \mathbb{A}\mathrm{gt}) \text{ such that } \boldsymbol{a}_i = \alpha_i \text{ for all } i \in C\}$$

denote the set of possible next states in CGS M when coalition C choose actions $\boldsymbol{a}_C$.

Given two concurrent game structures, $M_1 = (\mathbb{A}\text{gt}, St_1, Act_1, d_1, o_1, V_1)$ and $M_2 = (\mathbb{A}\text{gt}, St_2, Act_2, d_2, o_2, V_2)$, and a set of agents $C \subseteq \mathbb{A}\text{gt}$, a relation $\beta \subseteq St_1 \times St_2$ is a *C-bisimulation between M_1 and M_2*, denoted $M_1 \rightleftarrows^C_\beta M_2$, iff for all states $q_1 \in St_1$ and $q_2 \in St_2$, we have that $q_1 \beta q_2$ implies the following:

Local harmony $V_1(q_1) = V_2(q_2)$;
Forth For all joint actions $\boldsymbol{a}^1_C \in D_1(q_1, C)$ for C, there exists a joint action $\boldsymbol{a}^2_C \in D_2(q_2, C)$ for C such that for all states $q'_2 \in next_{M_2}(q_2, \boldsymbol{a}^2_C)$, there exists a state $q'_1 \in next_{M_1}(q_1, \boldsymbol{a}^1_C)$ such that $q'_1 \beta q'_2$;
Back Likewise, for 1 and 2 swapped.

A relation β is a *bisimulation between M_1 and M_2*, denoted $M_1 \rightleftarrows_\beta M_2$, if β is a C-bisimulation between M_1 and M_2 for every $C \subseteq \mathbb{A}\text{gt}$.

Using this notion of bisimulation, we can characterise the expressiveness of ATL as mentioned above. It is especially relevant when one wants to study the role of *memory*. Two different ways to define strategies are, first, as mappings from states to actions, and second, as mappings from sequences of states (histories) to actions. We call the first type *memoryless* strategies, modelling agents who base their actions on the current state only, and the second *perfect recall* strategies, modelling agents who base their strategies on the history of states. The semantics of ATL can be defined using either; by restricting the quantification ("there is a strategy..") in the semantic clauses to the appropriate class of strategies. To make the distinction formal, we will refer to the two semantics as $\models_{IR}, \models_{Ir}$ for "no **r**ecall" vs. "perfect **R**ecall" ("I" stands for "perfect **I**nformation", cf. Section 4.4 for more details).

We can now show that the language of "vanilla" ATL cannot discern between bisimilar CGSs (neither with memoryless nor with perfect recall strategies). In consequence, for "vanilla" ATL, memory does not matter: the interpretation of a formula (its truth value in a state of a structure) is the same whether we use only memoryless strategies or whether we use perfect recall strategies. On the other hand, memory matters for ATL*: the truth of a formula depends on the type of recall characterizing agents in the coalition.

Theorem 4 ([2]). *ATL_{Ir} is invariant under bisimulation. That is, if $M_1 \rightleftarrows_\beta M_2$ and $s_1 \beta s_2$, then for every ATL formula φ we have that:*

$$M_1, s_1 \models_{Ir} \varphi \quad \textit{iff} \quad M_2, s_2 \models_{Ir} \varphi.$$

The proof is somewhat technical; we refer to [2] for details.

Theorem 5 ([2]). *For "vanilla" ATL, memory does not matter. That is, $M, q \models_{Ir} \varphi$ iff $M, q \models_{IR} \varphi$.*

Proof (Sketch). Let $fincomp_M(q)$ denote the set of finite prefixes of computations starting in q. Let $\ell(q_0 \cdots q_k) = q_k$. Given a CGS $M = (\mathbb{A}\text{gt}, St, Act, d, o, V)$ and $q \in St$, the tree-unfolding $T(M, q)$ of M from q is defined as follows:

$$T(M, q) = (\mathbb{A}\text{gt}, St^*, Act, d^*, o^*, V^*),$$

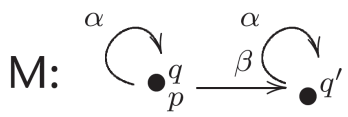

Fig. 5. Memory matters in ATL*: a concurrent game structure with single agent a

where $St^* = fincomp_M(q)$; $V^*(\sigma) = V(\ell(\sigma))$; $d_i^*(\sigma) = d_i(\ell(\sigma))$; and $o^*(\sigma, \mathbf{a}) = \sigma o(\ell(\sigma), \mathbf{a})$.

First, we observe that the tree unfolding includes exactly the same possibilities of action as the original model. Formally, for any M, q, we have $T(M,q) \rightleftarrows_\beta M$ where $\beta = \{(\sigma, \ell(\sigma)) \mid \sigma \in fincomp_M(q)\}$. Second, perfect recall strategies in M correspond exactly to memoryless the tree unfolding of M. Thus, $T(M,q), q \models_{Ir} \varphi \Leftrightarrow M, q \models_{IR} \varphi$ for every q and φ. Finally, by invariance under bisimulation, we obtain that $M, q \models_{Ir} \varphi$iff$T(M,q), q \models_{Ir} \varphi$ iff $M, q \models_{IR} \varphi$.

Corollary 1 ([2]). *ATL_{IR} is also invariant under bisimulation.*

Theorem 6 ([2]). *For ATL* memory matters. That is, there is a pointed model M, q and a formula φ such that the truth of φ in the model is different in the Ir and IR semantics.*

Proof. To see this, it suffices to consider the single-agent CGS M given in Figure 5 and $\varphi \equiv \langle\langle a \rangle\rangle(\mathrm{X}\mathsf{p} \wedge \mathrm{XX}\neg\mathsf{p})$.

References: Our presentation of ATL and its properties follows [4]. Meta-properties like axiomatization and model equivalence for ATL have been studied in [34, 2].

4.4 Strategic Reasoning for Imperfect Information

Bringing strategies and knowledge together: ATEL. ATL does not take into account the epistemic limitations of the agents; it assumes that every agent has *complete information* about the global state of the system. The *Alternating-time Temporal Epistemic Logic* ATEL was introduced in [98] as a straightforward combination of the multi-agent epistemic logic and ATL in order to formalize reasoning about the interaction of knowledge and abilities of agents and coalitions. ATEL enables specification of various modes and nuances of interaction between knowledge and strategic abilities, e.g.: $\langle\langle A \rangle\rangle\varphi \rightarrow \mathrm{E}_A\langle\langle A \rangle\rangle\varphi$ (if group A can bring about φ then everybody in A knows that they can), $\mathrm{E}_A\langle\langle A \rangle\rangle\varphi \wedge \neg\mathrm{C}_A\langle\langle A \rangle\rangle\varphi$ (the agents in A have mutual knowledge but not common knowledge that they can enforce φ); $\langle\langle i \rangle\rangle\varphi \rightarrow \mathrm{K}_i\neg\langle\langle \mathbb{A}\mathrm{gt} \setminus \{i\} \rangle\rangle\neg\varphi$ (if i can bring about φ then she knows that the rest of agents cannot prevent it), etc.

Models of ATEL are *concurrent epistemic game structures* (CEGS):

$$M = \langle \mathbb{A}\mathrm{gt}, St, Act, d, o, V, \sim_1, ..., \sim_k \rangle$$

combining the CGS-based models for ATL and the multi-agent epistemic models. The semantics of ATEL takes union of the semantic clauses for ATL and those from epistemic logic.

Problems with ATEL. While ATEL extends both ATL and epistemic logic, it also raises a number of conceptual problems. Most importantly, one would expect that an agent's ability to achieve property φ should imply that the agent has enough control and knowledge to *identify* and *execute* a strategy that enforces φ. A number of approaches have been proposed to overcome this problem. Most of the solutions agree that only *uniform* strategies (i.e., strategies that specify the same choices in indistinguishable states) are really executable. However, in order to identify a successful strategy, the agents must consider not only the courses of action, starting from the current state of the system, but also from states that are indistinguishable from the current one. There are many cases here, especially when group epistemics is concerned: the agents may have common, ordinary, or distributed knowledge about a strategy being successful, or they may be hinted the right strategy by a distinguished member (the "leader"), a subgroup ("headquarters committee") or even another group of agents ("consulting company").

Levels and modes of strategic ability. There are several possible interpretations of A's ability to bring about property γ, formalized by formula $\langle\langle A \rangle\rangle \gamma$, under imperfect information:

1. There exists behavior specification σ_A (not necessarily executable!) for agents in A such that, for every execution of σ_A, γ holds,
2. There is a uniform strategy s_A such that, for every execution of s_A, γ holds *(A have objective ability to enforce γ)*,
3. A know that there is a uniform s_A such that, for every execution of s_A, γ holds *(A have a strategy "de dicto" to enforce γ)*,
4. There is a uniform s_A such that A know that, for every execution of s_A, γ holds *(A have a strategy "de re" to enforce γ)*.

Note that the above interpretations form a sequence of increasingly stronger levels of ability – each next one implies the previous ones.

Case 4 is arguably most interesting, as it formalizes the notion of agents in A *knowing how to play*. However, the statement "A know that every execution of s_A satisfies γ" is precise only if A consists of a single agent a. Then, we take into account the paths starting from states indistinguishable from the current one according to a (i.e., $\bigcup_{q' \in \mathrm{img}(q, \sim_a)} out(q', s_a)$). In case of multiple agents, there are several different "modes" in which they can know the right strategy. That is, given strategy s, coalition A can have:

– *Common knowledge* that s is a winning strategy. This requires the least amount of additional communication when coordinating a joint strategy

(agents from A may agree upon a total order over their collective strategies at the beginning of the game and that they will always choose the maximal winning strategy with respect to this order);
- *Mutual knowledge* that s is a winning strategy: everybody in A knows that s is winning;
- *Distributed knowledge* that s is a winning strategy: if the agents share their knowledge at the current state, they can identify the strategy as winning;
- *"Leader"*: the strategy can be identified by an agent $a \in A$;
- *"Headquarters committee"*: the strategy can be identified by a subgroup $A' \subseteq A$;
- *"Consulting company"*: the strategy can be identified by another group B;
- ...other cases are also possible.

We will briefly present two variants of ATL that take into account the above considerations.

Economic solution: Schobbens' ATL$_{ir}$ and ATL$_{iR}$. In [88] a natural taxonomy of four strategy types was introduced and labeled as follows: I (resp. i) stands for *perfect* (resp. *imperfect*) *information*, and R (resp. r) refers to *perfect recall* (resp. *no recall*). The semantics of ATL can be parameterized with the strategy type – yielding four different semantic variants of the logic, labeled accordingly (ATL$_{IR}$, ATL$_{Ir}$, ATL$_{iR}$, and ATL$_{ir}$).

Like for ATEL, models are concurrent game structures augmented with a family of indistinguishability relations $\sim_a \subseteq St \times St$, one per agent $a \in \mathbb{A}\mathrm{gt}$. It is required that agents have the same choices in indistinguishable states: if $q \sim_a q'$ then $d(a,q) = d(a,q')$. We define two histories $h = q_0 q_1 \dots q_n$ and $h' = q'_0 q'_1 \dots q'_{n'}$ to be *indistinguishable for agent a under perfect recall* ($h \approx_a h'$) iff $n = n'$ and $q_i \sim_a q'_i$ for $i = 1, \dots, n$.

The following types of strategies are used in the respective semantic variants:

- Ir: $s_a : St \to Act$ such that $s_a(q) \in d(a,q)$ for all q;
- IR: $s_a : St^+ \to Act$ such that $s_a(q_0 \dots q_n) \in d(a,q_n)$ for all $q_0, \dots, q_n$;
- ir: like Ir, with the additional constraint that $q \sim_a q'$ implies $s_a(q) = s_a(q')$;
- iR: like IR, with the additional constraint that $h \approx_a h'$ implies $s_a(h) = s_a(h')$.

That is, strategy s_a is a conditional plan that specifies a's action in each state of the system (for memoryless agents) or for every possible history of the system evolution (for agents with perfect recall). Moreover, imperfect information strategies[4] specify the same choices for indistinguishable states (resp. histories). As before, collective xy-strategies s_A are tuples of individual xy-strategies s_a, one per $a \in A$. Note that the constraints in collective strategies refer to individual choices and individual relations $\sim_a$ (resp. $\approx_a$), and not to collective choices and any derived relations (e.g., $\sim_A^E$ and $\approx_A^E$).

We obtain the semantics for ATL$_{xy}$ by changing the clause for $\langle\langle A \rangle\rangle \gamma$ from Section 4.3 in the following way:

[4] We have already used the term *uniform* strategies exactly for this concept.

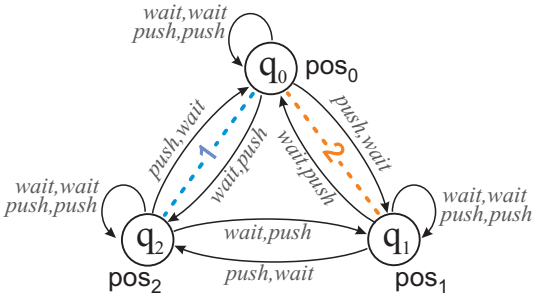

Fig. 6. Two robots and a carriage: imperfect information concurrent game structure M_5. Dashed lines represent indistinguishability relations between states.

$M, q \models_{xy} \langle\langle A \rangle\rangle \gamma$ iff there is an xy-strategy s_A for agents A such that for each path $\lambda \in \bigcup_{q' \in \mathrm{img}(q, \sim_A^E)} out(q', s_A)$, we have $M, \lambda \models_{xy} \gamma$.

Note that the above semantics of coalitional ability implements the "everybody knows" mode of identifying the strategy.

Example 6. Let us consider the concurrent epistemic game structure M_5 in Figure 6 that combines the strategic structure from model M_4 (Figure 4) with the epistemic relations from M_1 (Figure 1). Now, no agent knows how to make the carriage reach or avoid any selected state singlehandedly from q_0, i.e., $M_5, q_0 \models_{iy} \neg\langle\langle i \rangle\rangle \mathrm{F}\,\mathsf{pos_j}$ and $M_5, q_0 \models_{iy} \neg\langle\langle i \rangle\rangle \mathrm{G}\neg\mathsf{pos_j}$ for all $y \in \{\mathrm{r}, \mathrm{R}\}, i \in \{1, 2\}, j \in \{1, 2, 3\}$. Note in particular that the strategy from Example 5 cannot be used here because it is not uniform. The robots cannot even identify the right strategy together, e.g., $M_5, q_0 \models_{iy} \neg\langle\langle 1, 2 \rangle\rangle \mathrm{G}\neg\mathsf{pos_1}$ (when in q_0, robot 2 considers it possible that the system is already in the "bad" state q_1). So, do the robots know how to play to achieve anything? Yes, for example they know how to make the carriage reach a particular state eventually: $M_5, q_0 \models_{iy} \langle\langle 1, 2 \rangle\rangle \mathrm{F}\,\mathsf{pos_1}$ etc. – it suffices that one of the robots pushes all the time and the other waits all the time.

For the above properties the type of robots' recall does not matter (they hold in both memoryless and perfect recall strategies). $\langle\langle 1, 2 \rangle\rangle \mathrm{FG}\,\mathsf{pos_1}$ is an example ATL* formula that distinguishes between the two sets of strategies.

Constructive Strategic Logic. Most existing solutions (ATL_{ir} [88], ETSL [101]) treat only some of the cases (albeit in an elegant way), while others (ATOL [54], "Feasible ATEL" [55]) offer a more general treatment of the problem at the expense of an overblown logical language. *Constructive Strategic Logic* (*CSL*) [49], on the other hand, proposes a solution which we believe to be both intuitive, general and elegant. However, there is a price to pay here, too. In CSL, formulae are interpreted over *sets of states* rather than single states. We write $M, Q \models \langle\langle A \rangle\rangle \varphi$ to express the fact that A must have a strategy which is successful for all "opening" states from Q. New epistemic operators $\mathbb{K}_i, \mathbb{E}_A, \mathbb{C}_A, \mathbb{D}_A$ for "practical" or "constructive" knowledge yield the set of states for which a single evidence (i.e.,

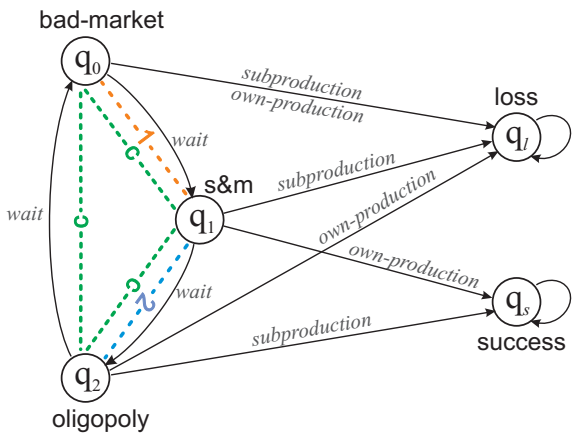

Fig. 7. Simple market: model M_6

a successful strategy) should be presented (instead of checking if the required property holds in each of the states separately, like standard epistemic operators do).

$M, Q \models \langle\langle A \rangle\rangle \varphi$ iff there is a uniform strategy s_A such that $M, \lambda \models \varphi$ for every $\lambda \in \cup_{q \in Q}\ out(q, s_A)$;
$M, Q \models \mathbb{K}_i \varphi$ iff $M, \{q' \mid \exists_{q \in Q}\ q \sim_i q'\} \models \varphi$;
$M, Q \models \mathbb{C}_A \varphi$ iff $M, \{q' \mid \exists_{q \in Q}\ q \sim_A^C q'\} \models \varphi$;
$M, Q \models \mathbb{E}_A \varphi$ iff $M, \{q' \mid \exists_{q \in Q}\ q \sim_A^E q'\} \models \varphi$;
$M, Q \models \mathbb{D}_A \varphi$ iff $M, \{q' \mid \exists_{q \in Q}\ q \sim_A^D q'\} \models \varphi$.

We point out that in CSL:

1. $\mathbb{K}_a \langle\langle a \rangle\rangle \varphi$ refers to agent a having a strategy *"de re"* to enforce φ (i.e. having a successful uniform strategy and knowing the strategy);
2. $K_a \langle\langle a \rangle\rangle \varphi$ refers to agent a having a strategy *"de dicto"* to enforce φ (i.e. knowing only that *some* successful uniform strategy is available);
3. $\langle\langle a \rangle\rangle \varphi$ expresses that agent a has a uniform strategy to enforce φ *from the current state* (but not necessarily even knows about it).

Thus, $\mathbb{K}_i \langle\langle i \rangle\rangle \varphi$ captures the notion of i's knowing *how to play* to achieve φ, while $K_i \langle\langle i \rangle\rangle \varphi$ refers to knowing only *that a successful play is possible*. This extends naturally to abilities of coalitions.

Example 7 (Market scenario). Consider an industrial company that wants to start production, and looks for a good strategy when and how it should do it. The market model is depicted in Figure 7. The economy is assumed to run in simple cycles: after the moment of bad economy (bad-market), there is always a good time for small and medium enterprises (s&m), after which the market

tightens and an oligopoly emerges. At the end, the market gets stale, and we have stagnation and bad economy again.

The company c is the only agent whose actions are represented in the model. The company can wait (action *wait*) or decide to start production: either on its own (*own-production*), or as a subcontractor of a major company (*subproduction*). Both decisions can lead to either loss or success, depending on the current market conditions. However, the company management cannot recognize the market conditions: bad market, time for small and medium enterprises, and oligopoly market look the same to them, as the epistemic links for c indicate.

The company can call the services of two marketing experts. Expert 1 is a specialist on oligopoly, and can recognize oligopoly conditions (although she cannot distinguish between bad economy and s&m market). Expert 2 can recognize bad economy, but he cannot distinguish between other types of market. The experts' actions have no influence on the actual transitions of the model, and are omitted from the graph in Figure 7. It is easy to see that the company cannot identify a successful strategy on its own: for instance, for the small and medium enterprises period, we have that $M_6, q_1 \models \neg\mathbb{K}_c\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$. It is not even enough to call the help of a single expert: $M_6, q_1 \models \neg\mathbb{K}_1\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success} \wedge \neg\mathbb{K}_2\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$, or to ask the experts to independently work out a common strategy: $M_6, q_1 \models \neg\mathbb{E}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$. Still, the experts can propose the right strategy if they join forces and cooperate to find the solution: $M_6, q_1 \models \mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$.

This is not true any more for bad market, i.e., $M_6, q_0 \models \neg\mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$, because c is a memoryless agent, and it has no uniform strategy to enforce $\mathsf{success}$ from q_0 at all. However, the experts can suggest a more complex scheme that involves consulting them once again in the future, as evidenced by $M_6, q_0 \models \mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{X}\ \mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$.

References: Imperfect information variants of ATL have been studied in numerous papers. We recomend especially [88, 49]. The former clarifies the basic conceptual structure of what it means to have strategic ability for some temporal property. The latter give a more involved treatment of the interplay between the epistemic and the strategic dimensions.

5 Verification of Strategic Abilities

We present an overview of complexity results for model checking strategic logic. The survey is based on the overview papers [47] and especially [9].

5.1 Model Checking ATL and CL: Perfect Information

One of the main results concerning ATL states that its formulae can also be model-checked in deterministic linear time, analogously to CTL. It is important to emphasize, however, that the result is relative to the number of transitions in the model and the length of the formula.

The ATL model checking algorithm from [4] is presented in Figure 8. The algorithm employs the well-known fixpoint characterizations of strategic-temporal modalities:

$$\langle\langle A\rangle\rangle \mathrm{G}\varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle \mathrm{X}\langle\langle A\rangle\rangle \mathrm{G}\varphi$$
$$\langle\langle A\rangle\rangle \varphi_1 \mathrm{U} \varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \langle\langle A\rangle\rangle \mathrm{X}\langle\langle A\rangle\rangle \varphi_1 \mathrm{U} \varphi_2,$$

and computes a winning strategy step by step (if it exists). That is, the algorithm starts with the appropriate candidate set of states ($\emptyset$ for U and the whole set St for G), and iterates backwards over A's one-step abilities until the set gets stable. It is easy to see that the algorithm needs to traverse each transition at most once per subformula of φ. Note that it does not matter whether perfect recall or memoryless strategies are used: the algorithm is correct for the *IR*-semantics, but it always finds an *Ir*-strategy. Thus, for an ATL-formula $\langle\langle A\rangle\rangle\gamma$, if A have an *IR*-strategy to enforce γ, they also have an *Ir*-strategy to obtain it.

Theorem 7 ($\mathbf{ATL}_{Ir}$ and $\mathbf{ATL}_{IR}$ [4]). *Model checking ATL_{Ir} and ATL_{IR} is* **P***-complete, and can be done in time* $\mathbf{O}(|M| \cdot |\varphi|)$*, where $|M|$ is given by the number of transitions in M.*

Proof (Sketch). Each case of the algorithm is called at most $\mathbf{O}(|\varphi|)$ times and terminates after $\mathbf{O}(|M|)$ steps [4]. The latter is shown by translating the model to a two-player game [4], and then solving the "invariance game" on it in polynomial time [5]. Hardness is shown by a reduction of reachability in And-Or-Graphs, which was shown to be **P**-complete in [44], to model checking the (constant) ATL-formula $\langle\langle 1\rangle\rangle \mathrm{F}\mathtt{p}$ in a two player game. In each Or-state it is the turn of player 1 and in each And-state it is player 2's turn [4]. □

The next theorem shows that model checking of coalition logic is as hard as for ATL.

Theorem 8 ($\mathbf{CL}_{Ir}$ and $\mathbf{CL}_{IR}$ [9]). *Model checking CL_{Ir} and CL_{IR} is* **P***-complete, and can be done in time* $\mathbf{O}(|M| \cdot |\varphi|)$*, where $|M|$ is given by the number of transitions in M.*

function $mcheck(M, \varphi)$**.**

ATL model checking. Returns the set of states in model $M = \langle \mathbb{A}\mathrm{gt}, St, V, o\rangle$ for which formula φ holds.

case $\varphi \in \mathcal{PV}$ **:** return $V(p)$
case $\varphi = \neg\psi$ **:** return $St \setminus mcheck(M, \psi)$
case $\varphi = \psi_1 \vee \psi_2$ **:** return $mcheck(M, \psi_1) \cup mcheck(M, \psi_2)$
case $\varphi = \langle\langle A\rangle\rangle \mathrm{X}\psi$ **:** return $pre(M, A, mcheck(M, \psi))$
case $\varphi = \langle\langle A\rangle\rangle \mathrm{G}\psi$ **:**
  $Q_1 := St$; $Q_2 := mcheck(M, \psi)$; $Q_3 := Q_2$;
  while $Q_1 \not\subseteq Q_2$
  do $Q_1 := Q_2$; $Q_2 := pre(M, A, Q_1) \cap Q_3$ **od**;
  return Q_1
case $\varphi = \langle\langle A\rangle\rangle \psi_1 \mathrm{U} \psi_2$ **:**
  $Q_1 := \varnothing$; $Q_2 := mcheck(M, \psi_1)$;
  $Q_3 := mcheck(M, \psi_2)$;
  while $Q_3 \not\subseteq Q_1$
  do $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre(M, A, Q_1) \cap Q_2$ **od**;
  return Q_1
end case

function $pre(M, A, Q)$**.**

Auxiliary function; returns the exact set of states Q' such that, when the system is in a state $q \in Q'$, agents A can cooperate and enforce the next state to be in Q.

return $\{q \mid \exists \alpha_A \forall \alpha_{\mathbb{A}\mathrm{gt}\setminus A}\ o(q, \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in Q\}$

Fig. 8. The ATL model checking algorithm from [4]

Proof. The upper bound follows from the fact that CL is a sublanguage of ATL. We show **P**-hardness by the following adaption of the reduction of And-Or-Graph reachability from [4]. Firstly, we observe that if a state y is reachable from x in graph G then it is also reachable via a path whose length is bounded by the number n of states in the graph. Like in the proof of Theorem 7, we take G to be a turn-based CGS in which player 1 "owns" all the Or-states and player 2 "owns" all the And-states. We also label node y with a special proposition y, and replace all the transitions outgoing from y with a deterministic loop. Now, we have that y is reachable from x in G iff $G, x \models \underbrace{\langle\langle 1\rangle\rangle \mathrm{X} \ldots \langle\langle 1\rangle\rangle \mathrm{X}}_{n\text{-times}} \mathsf{y}$. The reduction uses only logarithmic space. □

It is worth pointing out, however, that checking strategic properties in one-step games is somewhat easier. We recall that $\mathbf{AC^0}$ is the class corresponding to constant-depth, unbounded-fanin, polynomial-size Boolean circuits with AND, OR, and NOT gates [31]. We call a formula *flat* if it contains no nested cooperation modalities. Moreover, a formula is *simple* if it is flat and does not include Boolean connectives. For example, the language of "simple CL" consists only of formulae p and $\langle\langle A\rangle\rangle \mathrm{X}p$, for $p \in \mathcal{PV}$ and $A \subseteq \mathbb{A}\mathrm{gt}$.

Theorem 9 (Simple $\mathbf{CL}_{Ir}$ and $\mathbf{CL}_{IR}$ [64]). *Model checking "Simple CL_{Ir}" and "Simple CL_{IR}" with respect to the number of transitions in the model and the length of the formula is in* $\mathbf{AC^0}$.

Proof (Sketch). For $M, q \models \langle\langle A \rangle\rangle \mathsf{X}\mathsf{p}$, we construct a 3-level circuit [64]. On the first level, we assign one AND gate for every possible coalition B and B's collective choice α_B.[5] The output of the gate is "true" iff α_B leads to a state satisfying p for every response of $\mathbb{A}\mathrm{gt} \setminus B$. On the second level, there is one OR gate per possible coalition B that connects all the B's gates from the first level and outputs "true" iff there is any successful strategy for B. On the third level, there is a single AND gate that selects the right output (i.e., the one for coalition A). □

5.2 Model Checking ATL and CL: Imperfect Information

In contrast to the perfect information setting, analogous fixpoint characterizations need *not* hold for the incomplete information semantics over ATL. This is because the choice of a particular action at a state q has non-local consequences: it automatically fixes agent i's choices at all states q' indistinguishable from q for i. Note that, for two different members of coalition A, uniformity of their parts of the coalitional strategy imposes different constraints on their choices if their epistemic relations are not exactly the same. Moreover, the agents' ability to *identify* a strategy as winning also varies throughout the game in an arbitrary way (agents can learn as well as forget). This suggests that winning strategies cannot be synthesized incrementally. Note that, in order to check $M, q \models \langle\langle A \rangle\rangle \gamma$ (where γ includes no nested cooperation modalities), the following procedure suffices. Firstly, we guess a uniform strategy s_A of team A (by calling an **NP** oracle), and then verify the strategy by pruning M accordingly (removing all the transitions that are not going to be executed according to s_A) and model-checking the CTL-formula $\mathsf{A}\gamma$ in the resulting model. For nested cooperation modalities, we proceed recursively (bottom up). Since model checking CTL can be done in polynomial deterministic time, the procedure runs in polynomial deterministic time with calls to an **NP** oracle, which demonstrates the inclusion in $\mathbf{\Delta_2^P} = \mathbf{P^{NP}}$ [88]. As it turns out, a more efficient procedure does not exist, which is confirmed by the following result.

Theorem 10 ($\mathbf{ATL}_{ir}$ [88, 53]). *Model checking ATL_{ir} is $\mathbf{\Delta_2^P}$-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). The discussion above proves the membership in $\mathbf{\Delta_2^P}$. $\mathbf{\Delta_2^P}$-hardness was shown in [53] through a reduction of *sequential satisfiability* ($\mathbf{SNSAT}_2$), a standard $\mathbf{\Delta_2^P}$-complete problem [65]. The idea is that there are two agents where one agent tries to verify a (nested) propositional formula and a second agent tries to refute it. A winning strategy of the "verifier agent" corresponds to a satisfying

[5] The number of coalitions is exponential in the number of agents, but polynomial in the size of the input (provided that $p \geq 2$) because there are at least $2^{|\mathbb{A}\mathrm{gt}|}$ transitions in the model.

function *mcheck*(M, q, φ)**.**

Model checking CL formulae of type $\varphi \equiv \langle\langle a_1, a_2 \rangle\rangle \mathrm{X}\mathsf{p}$.

Let $Q = [q]_{\sim_A}$ and $D = d_{a_1}(q) \times d_{a_2}(q)$;
while there is still a collective action (α_1, α_2) in D **do**:

- fix α_1 for a_1 in $[q]_{\sim_{a_1}}$ and α_2 for a_2 in $[q]_{\sim_{a_2}}$.
- for every state in $[q]_{\sim_A}$ there is at most one agent in A for whom the action has not been fixed. **if** a_i's action is not fixed for q', q'' such that $q' \sim_{a_i} q''$ **then** collapse q', q'' into a single state (taking the union of the outgoing transitions). **repeat** iteratively;
- **if** in the resulting perfect information CEGS A have a one-step strategy to enforce p in the next state **then** return *true* **else** remove (α_1, α_2) from D and revert to the original model M;

if the loop ended with no success (i.e., there are no more available actions) **then** return *false*.

Fig. 9. Model checking Coalition Logic for small teams and imperfect information

valuation of the formula. Uniformity of the verifier's strategy is needed to ensure that identical proposition symbols, occurring at different places in the formula, are assigned the same truth values. □

Now we consider the incomplete information setting for Coalition Logic. It is easy to see that the *iR*- and *ir*-sematics are equivalent for CL since X is the only temporal operator, and thus only the first action in a strategy matters. As a consequence, whenever there is a successful *iR*-strategy for agents A to enforce $\mathrm{X}\varphi$, then there is also an *ir*-strategy for A to obtain the same. Perfect recall of the history does not matter in one-step games. Surprisingly, what matters is the size of teams that are allowed to cooperate.

Theorem 11 ($\mathbf{CL_{ir}}$ and $\mathbf{CL_{iR}}$ [11]). *Model checking CL_{ir} and CL_{iR} for formulae that include only strategic operators $\langle\langle A \rangle\rangle$ with $|A| \leq 2$ is* **P***-complete wrt the number of transitions in the model and the length of the formula, and can be done in time* $\mathbf{O}(|M| \cdot |\varphi|)$.

Proof. The **P**-hardness follows from Theorem 8 (perfect information CGS's can be seen as a special kind of CEGS where the indistinguishability relations contain only the reflexive loops).

To obtain the upper bound, we use the algorithm in Figure 9 to model check $M, q \models \langle\langle A \rangle\rangle \mathrm{X}\mathsf{p}$ (with $A = \{a_1, a_2\}$). It is easy to see that the algorithm never processes the same transition twice. For $\langle\langle A \rangle\rangle \mathrm{X}\varphi$ with nested cooperation modalities, we proceed recursively (bottom up). □

Theorem 12 ($\mathbf{CL_{ir}}$ and $\mathbf{CL_{iR}}$ [11]). *Model checking CL_{ir} and CL_{iR} for $|A| \geq 3$ is between* **NP** *and* $\mathbf{\Delta_2^P}$ *wrt the number of transitions in the model and the length of the formula. It is conjectured to be* $\mathbf{\Delta_2^P}$*-complete.*

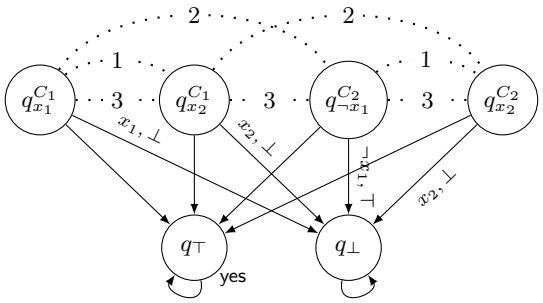

Fig. 10. Model M_Φ for $\Phi \equiv C_1 \wedge C_2, C_1 \equiv x_1 \vee x_2, C_2 \equiv \neg x_1 \vee x_2$. Only transitions leading to $q_\bot$ are labeled; the other combinations of actions lead to $q_\top$.

Proof. The upper bound is obtained by the following algorithm. To check $M, q \models \langle\langle A \rangle\rangle \mathsf{X}\mathsf{p}$, we guess a one-step strategy of A, remove from M the irrelevant transitions and states outside $[q]_{\sim_A}$, and check if p holds for all the remaining "next" states. For $\langle\langle A \rangle\rangle \mathrm{X}\varphi$ with nested cooperation modalities, we proceed recursively (bottom up).

For the lower bound, we use a reduction of the Boolean satisfiability problem (SAT). Given a Boolean formula Φ in CNF, we construct a 3-agent CEGS M_Φ as follows. Each literal l from clause ψ in Φ is associated with a state q_l^ψ. At state q_l^ψ, player 1 indicates a literal from ψ, and player 2 decides on the valuation of the underlying Boolean variable. If 1 indicated a "wrong" literal $l' \neq l$ then the system proceeds to state $q_\top$ where proposition yes holds. The same happens if 1 indicated the "right" literal (l) and 2 selected the valuation that makes l true. Otherwise the system proceeds to the "sink" state $q_\bot$.

Player 1 must select literals uniformly within clauses, so $q_l^\psi \sim_1 q_{l'}^{\psi'}$ iff $\psi = \psi'$. Player 2 is to select uniform valuations of variables, i.e., $q_l^\psi \sim_2 q_{l'}^{\psi'}$ iff $var(l) = var(l')$ where $var(l)$ is the variable contained in l. Finally, all states except $q_\top, q_\bot$ are indistinguishable for 3. An example of the construction is presented in Figure 10.

Then, Φ is satisfiable iff $M_\Phi, q \models \langle\langle 1,2,3 \rangle\rangle \mathsf{X}\mathsf{yes}$ where q is an arbitrary "literal" state. □

The last possibility for defining the semantics of ATL is to assume imperfect information and perfect recall of agents. The problem has been recently proved undecidable in [21].

Theorem 13 ($\mathbf{ATL_{iR}}$ [21]). *Model checking ATL_{iR} is undecidable.*

Proof (Sketch). The proof follows by a reduction of the non-halting problem for nondeterministic Turing machines. Given a machine T, we construct a CEGS M such that the evolution of the configuration of T is simulated by the tree

of computations in M. That is, subsequent configurations of T are represented by subsequent levels in the tree unfolding of M. There are 3 agents in M: the proponents (agents 1 and 2) who "move" the head of the tape in T, and the opponent (agent 3) who takes care of splitting the branches (i.e., adding new symbols to the tape whenever they are needed). The indistinguishability relations for 1 and 2 are constructed so that moving the head (and possibly changing the current state of T) proceeds uniformly; 1 takes care of the left-hand side of the head, and 2 of the right-hand side. Special proposition ok labels elements of configurations from which T has to proceed further. Thus, we get that T does not terminate iff $M, q_0 \models \langle\langle 1,2 \rangle\rangle \mathrm{G}\mathsf{ok}$.

For more details, we refer to the technical construction in [21]. □

Remark 1. The theorem is well aligned with the tradition of previous undecidability results [78, 77, 80, 100, 91, 90]. However, contrary to what most people had thought, it did not follow from those results.

In the papers [78, 77], solving games with imperfect information, perfect recall and multiple proponents is proved undecidable, but those games are defined by Turing machines. In case of ATL_{iR} models can be seen as Büchi automata with simple acceptance conditions (reachability or safety). We note that, for games defined by Turing machines, more sophisticated "winning conditions" can be specified, e.g., we can imagine a game in which the protagonist agents must count specific actions of the opponent in order to win. As a more precise example, imagine a game in which team $T_1 = \{a_1, a_2\}$ is playing against the "environment" agent a_0 in the following way: first, the opponent (a_0) throws in any finite number of $\bigcirc$ symbols onto the tape, and then agents a_1, a_2 can write a number of #'s, before they decide to "call". After that, a_0 checks if the amounts of $\bigcirc$'s and #'s are the same: if so, team T_1 wins, otherwise it loses the game. Clearly, such a winning condition cannot be specified through a CEGS model with a subset of states marked as "winning" states, because the language $\{\bigcirc^n \#^n \epsilon^\omega \mid n \in \mathbb{N}\}$ is not ω-regular. Moreover, using a Turing Machine allows to define more sophisticated rules of how the game proceeds, e.g. the protagonists can be allowed to put another symbol $\square$ on the tape only when they have already put as many #'s as there are $\bigcirc$'s left by the opposition.

The paper [80] shows that that the *LTL realizability problem* for distributed systems is undecidable. In fact, this implies that model checking of ATL^*_{iR} is undecidable, but it does not carry over to "vanilla" ATL_{iR}. This is because "winning conditions" in [80] are defined through LTL specifications, so we have winning *paths* rather than states. Moreover, the reduction of the halting problem to the realizability problem employs LTL formulae that are not expressible in CTL and ATL (cf. [27]).

Finally, the paper [100] shows undecidability of model checking LTL with perfect recall and common knowledge, and [91, 90] gives analogous results for CTL. However, ATL_{iR} does not allow for expressing common knowledge with perfect recall.

It is worth pointing out that for single-agent coalitions the model checking problem is decidable – more precisely, EXPTIME-complete [22]. The decidabil-

ity follows also from a more general result in [35] stating that model checking abilities of coalitions with unrestricted communication is decidable.

5.3 Model Checking ATL*

We now turn to model checking for the broader logic ATL*.

Theorem 14 ($\mathbf{ATL^*_{IR}}$ [4]). *Model checking ATL^*_{IR} is* **2EXPTIME***-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). Let M be a CGS and $\langle\langle A\rangle\rangle\psi$ be an ATL*-formula (where we assume that ψ is an LTL-formula). Given a strategy s_A of A and a state q in M the model can be unfolded into a q-rooted tree representing all possible behaviors with agents A following their strategy s_A. This structure can be seen as the tree *induced* by $out(q, s_A)$ and we will refer to it as a (q, A)-*execution tree*. Note that every strategy profile for A may result in a different execution tree. Now, a Büchi tree automaton $\mathcal{A}_{M,q,A}$ can be constructed that accepts exactly the (q, A)-execution trees [4].

Secondly, it was shown that one can construct a Rabin tree automaton which accepts all trees that satisfy the CTL*-formula $\mathsf{A}\psi$ [26]. Hence, the ATL*-formula $\langle\langle A\rangle\rangle\psi$ is satisfied in M, q if there is a tree accepted by $\mathcal{A}_{M,q,A}$ (i.e., it is a (q, A)-execution tree) and by $\mathcal{A}_\psi$ (i.e., it is a model of $\mathsf{A}\psi$).

The lower bound is shown by a reduction of the **2EXPTIME**-complete problem of the realizability of LTL-formulae [79, 85, 4]. □

On the other hand, model checking ATL* *with memoryless strategies* is no worse than for LTL and CTL* with perfect as well as imperfect information.

Theorem 15 ($\mathbf{ATL^*_{ir}}$ and $\mathbf{ATL^*_{Ir}}$ [88]). *Model checking ATL^*_{ir} and ATL^*_{Ir} is* **PSPACE***-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). ATL* embeds LTL (every LTL formula φ is equivalent to the ATL* formula $\langle\langle\emptyset\rangle\rangle\varphi$) which renders model checking ATL* with memoryless strategies to be **PSPACE**-hard.

On the other hand, there is a **PSPACE** algorithm for model checking ATL* with memoryless strategies (for perfect as well as imperfect information). Consider the formula $\langle\langle A\rangle\rangle\psi$ where ψ is an LTL-formula. Then, an *ir*-strategy s_A for A is guessed and the model is "trimmed" according to s_A, i.e. all transitions which cannot occur by following s_A are removed. Note that a memoryless strategy can be guessed in polynomially many steps, and hence also using only polynomially many memory cells. In the new model the CTL*-formula $\mathsf{A}\psi$ is checked. This procedure can be performed in $\mathbf{NP}^{\mathbf{PSPACE}}$, which renders the complexity of the whole language to be in $\mathbf{P}^{\mathbf{NP}^{\mathbf{PSPACE}}} = \mathbf{PSPACE}$. □

The following is an immediate consequence of Theorem 13.

Theorem 16 ($\mathbf{ATL^*_{iR}}$). *Model checking ATL^*_{iR} is undecidable.*

Figure 11 presents an overview of the model checking complexity results for explicit models.

	Ir	*IR*	*ir*	*iR*
CL	**P**	**P**	$\mathbf{P}/\mathbf{\Delta_2^P}$ †	$\mathbf{P}/\mathbf{\Delta_2^P}$ †
ATL	**P**	**P**	$\mathbf{\Delta_2^P}$	Undecidable
ATL*	**PSPACE**	**2EXPTIME**	**PSPACE**	Undecidable

Fig. 11. Overview of the model checking complexity results for explicit models. All results except for "Simple CL" are completeness results. Each cell represents the logic over the language given in the row using the semantics given in the column. † The problem is **P**-complete for formulae including only small coalitions (of at most 2 agents) and between **NP** and $\mathbf{\Delta_2^P}$ in general. We conjecture that it is $\mathbf{\Delta_2^P}$-complete in the latter case.

5.4 Complexity for Compact Representation of Transitions

In this section we consider the complexity of the model checking problem *with respect to the number of states, agents, and an implicitly encoded transition function* rather than the (explicit) number of transitions. It is easy to see that, for CGS's, the number of transitions can be exponential in the number of states and agents. Therefore, all the algorithms presented in Sections 5.1-5.3 give us only exponential time bounds provided that the encoding of the transition function is sufficiently small.

Proposition 1 ([51, 4]). *Let n be the number of states in a concurrent game structure M, let k denote the number of agents, and d the maximal number of available decisions (moves) per agent per state. Then, the number of transitions $m = \mathbf{O}(nd^k)$. Therefore the ATL_{IR}/ATL_{Ir} model checking algorithm from [4] runs in time $\mathbf{O}(nd^k l)$ where l is the length of the formula, and hence its complexity is exponential if the number of agents is a parameter of the problem.*

In comparison, for an *unlabeled* transition system with n states and m transitions, we have that $m = \mathbf{O}(n^2)$. This means that CTL model checking is in **P** also with respect to the number of states in the model and the length of the formula. The following theorem is an immediate corollary of the fact (and Theorem 1).

Theorem 17. *CTL model checking over unlabeled transition systems is* **P***-complete in the number of states and the length of the formula, and can be done in time* $\mathbf{O}(n^2 l)$.

For ATL and concurrent game structures, however, the situation is different. In the following we make precise what we mean by a compressed transition function.

Implicit concurrent game structures (called this way first in [64], but already present in the ISPL modeling language behind MCMAS [82, 81]) are defined similarly to CGS's but the transition function is encoded in a more compact way by a sequence

$$((\varphi_0^r, q_0^r), \ldots, (\varphi_{t_r}^r, q_{t_r}^r))_{r=1,\ldots,|Q|}$$

where $t_r \in \mathbb{N}_0$, $q_i^r \in St$ and each φ_i^r is a Boolean combination of propositions exec_α^j where $j \in \mathbb{A}\mathrm{gt}$, $\alpha \in Act$, $i = 1, \ldots, t$ and $r = 1, \ldots, |Q|$. It is required that $\varphi_{t_r}^r = \mathbf{true}$. The term exec_α^j stands for "agent j executes action α". We use $\varphi[\alpha_1, \ldots, \alpha_k]$ to refer to the Boolean formula over $\{\mathbf{true}, \mathbf{false}\}$ obtained by replacing $\mathsf{exec}_\alpha^{\mathsf{a}j}$ with $\mathbf{true}$ (resp. $\mathbf{false}$) if $\alpha_j = \alpha$ (resp. $\alpha_j \neq \alpha$). The encoding defines a transition function $\hat{o}$ as follows:

$$\hat{o}(q_i, \alpha_1, \ldots, \alpha_k) = q_j^i \text{ where } j = \min\{\kappa \mid \varphi_\kappa^i[\alpha_1, \ldots, \alpha_k] \equiv \mathbf{true}\}$$

That is, $\hat{o}(q_i, \alpha_1, \ldots, \alpha_k)$ returns the state belonging to the formula φ_κ^i (associated with state q_i) with the minimal index κ that evaluates to "true" given the actions $\alpha_1, \ldots, \alpha_k$. Note that the function is well defined as the last formula in each sequence is given by $\mathbf{true}$: no deadlock can occur. The size of $\hat{o}$ is defined as $|\hat{o}| = \sum_{r=1,\ldots,|Q|} \sum_{j=1,\ldots,t_r} |\varphi_j^r|$, that is, the sum of the sizes of all formulae. Hence, the size of an implicit CGS is given by $|St| + |\mathbb{A}\mathrm{gt}| + |\hat{o}|$. Recall that the size of an explicit CGS is $|St| + |\mathbb{A}\mathrm{gt}| + m$ where m is the number of transitions.

Remark 2. Note that if the transition function is given by an array that *explicitly enumerates all transitions* then the complexity is trivially linear (or even lower) *in the size of the input* which includes the representation of the transition function. However, enumerating the exponentially many transitions is not feasible in all but the simplest models.

Theorem 18 ([64, 51, 53]). *Model checking ATL_{IR} and ATL_{Ir} over implicit CGS's is $\mathbf{\Delta_3^P}$-complete with respect to the size of the model (defined by the number of states, agents, and implicit transitions) and the length of the formula.*

Proof (Sketch). The idea of the proof for the lower bound is clear if we reformulate the model checking of $M, q \models \langle\langle a_1, \ldots, a_r \rangle\rangle \mathrm{X}\varphi$ as

$$\exists(\alpha_1, \ldots, \alpha_r) \forall(\alpha_{r+1}, \ldots, \alpha_k)\ M, o(q, \alpha_1, \ldots, \alpha_k) \models \varphi,$$

which closely resembles QSAT_2, a typical $\mathbf{\Sigma_2^P}$-complete problem. A reduction of this problem to our model checking problem is straightforward. For each instance of QSAT_2, we create a model where the values of propositional variables $p_1, \ldots, p_r$ are "declared" by agents A and the values of $p_{r+1}, \ldots, p_k$ by $\mathbb{A}\mathrm{gt} \setminus A$. The subsequent transition leads to a state labeled by proposition yes iff the given Boolean formula holds for the underlying valuation of $p_1, \ldots, p_k$. Then, QSAT_2 reduces to model checking formula $\langle\langle a_1, \ldots, a_r \rangle\rangle \mathrm{X}\mathsf{yes}$ [51]. In order to obtain $\mathbf{\Delta_3^P}$-hardness, the above schema is combined with nested cooperation modalities, which yields a rather technical reduction of the SNSAT_3 problem that can be found in [64].

For the upper bound, we consider the following algorithm for checking $M, q \models \langle\langle A \rangle\rangle \gamma$ with no nested cooperation modalities. Firstly, guess a strategy s_A of the proponents and fix A's actions to the ones described by s_A. Then check if $\mathsf{A}\gamma$ is true in state q of the resulting model by asking an oracle about the existence of a counterstrategy $s_{\bar{A}}$ for $\mathbb{A}\mathrm{gt} \setminus A$ that falsifies γ and reverting the oracle's

answer. The evaluation takes place by calculating $\hat{o}$ (which takes polynomially many steps) regarding the actions prescribed by $(s_A, s_{\bar{A}})$ at most $|St|$ times. For nested cooperation modalities, we proceed recursively (bottom-up). □

Surprisingly, the imperfect information variant of ATL is no harder than the perfect information one under this measure:

Theorem 19 ([53]). *Model checking ATL_{ir} over implicit CGS's is $\mathbf{\Delta_3^P}$-complete with respect to the size of the model and the length of the formula. This is the same complexity as for model checking ATL_{Ir} and ATL_{IR}.*

Proof (Sketch). For the upper bound, we use the same algorithm as in checking ATL_{Ir}. For the lower bound, we observe that ATL_{Ir} can be embedded in ATL_{ir} by explicitly assuming perfect information of agents (through the minimal reflexive indistinguishability relations). □

The $\mathbf{\Delta_3^P}$-hardness proof in Theorem 18 uses the "nexttime" and "until" temporal operators in the construction of an ATL formula that simulates SNSAT_3 [64]. However, the proof can be modified so that only the "nexttime" sublanguage of ATL is used. We obtain thus an analogous result for coalition logic. Details of the new construction can be found in [8].

Theorem 20. *Model checking CL_{IR}, CL_{Ir}, CL_{ir}, and CL_{iR} over implicit CGS's is $\mathbf{\Delta_3^P}$-complete with respect to the size of the model and the length of the formula.*

It is worth mentioning that model checking "Positive ATL" (i.e., the fragment of ATL where negation is allowed only on the level of literals) is $\mathbf{\Sigma_2^P}$-complete with respect to the size of implicit CGS's, and the length of formulae for the *IR*, *Ir*, and *ir*-semantics [53]. The same applies to "Positive CL", the analogous variant of coalition logic.

5.5 CTL and ATL* Revisited

Theorem 21 ([9]). *Model checking CTL over implicit CGS's is $\mathbf{\Delta_2^P}$-complete with respect to the size of the model and the length of the formula.*

Proof (Sketch). For the upper bound, we observe that $M, q \models_{CTL} \mathsf{E}\gamma$ iff $M, q \models_{ATL_{IR}} \langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \gamma$ which is in turn equivalent to $M, q \models_{ATL_{Ir}} \langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \gamma$. In other words, $\mathsf{E}\gamma$ holds iff the grand coalition has a *memoryless* strategy to achieve γ. Thus, we can verify $M, q \models_{CTL} \mathsf{E}\gamma$ (with no nested path quantifiers) as follows: we guess a strategy $s_{\mathbb{A}\mathrm{gt}}$ for $\mathbb{A}\mathrm{gt}$ (in polynomially many steps), then we construct the resulting model M' by asking $\hat{o}$ which transitions are enabled by following the strategy s_A and check if $M', q \models_{CTL} \mathsf{E}\gamma$ and return the answer. Note that M' is an *unlabeled transition system*, so constructing M' and checking $M', q \models_{CTL} \mathsf{E}\gamma$ can be done in polynomial time. For nested modalities, we proceed recursively.

For the lower bound, we sketch a reduction of SAT to model checking CTL-formulae with only one path quantifier. For propositional variables $p_1, \ldots, p_k$ and boolean formula φ, we construct an implicit CGS where the values of $p_1, \ldots, p_k$

	Ir	*IR*	*ir*	*iR*
CL	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$
ATL	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	Undecidable
ATL*	**PSPACE**	**2EXPTIME**	**PSPACE**	Undecidable

Fig. 12. Overview of the model checking complexity results for implicit CGS. All results are completeness results. Each cell represents the logic over the language given in the row using the semantics given in the column.

are "declared" by agents $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ (in parallel). The subsequent transition leads to a state labeled by proposition yes iff φ holds for the underlying valuation of $p_1, \ldots, p_k$. Then, SAT reduces to model checking formula $\langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \mathrm{X}\mathsf{yes}$. The reduction of SNSAT$_2$ (to model checking CTL-formulae with nested path quantifiers) is an extension of the SAT reduction, analogous to the one in [52, 53]. □

Theorem 22. *Model checking ATL^*_{Ir} and ATL^*_{ir} over implicit CGS's is* **PSPACE***-complete with respect to the size of the model and the length of the formula.*

Proof. We observe that every explicit CGS can be encoded as an implicit CGS with no blowup in size. In consequence, the lower bound follows from Theorem 15.

For the upper bound, we model-check $M, q \models \langle\langle A \rangle\rangle \gamma$ by guessing a memoryless strategy s_A for coalition A, then we guess a perfect information memoryless counterstrategy $s_{\bar{A}}$ of the opponents. Having a complete strategy profile, we proceed as in the proof of Theorem 21 and check the LTL path formula γ on the resulting (polynomial model) M' which can be done in polynomial space (Theorem 15). For nested cooperation modalities, we proceed recursively. □

Theorem 23 ([64]). *Model checking ATL^*_{IR} over implicit CGS's is* **2EXPTIME***-complete with respect to the size of the model and the length of the formula.*

Proof. Again, the lower bound follows from Theorem 14. For the upper bound, we have to modify the algorithm given in the proof of Theorem 14 so that it is capable of dealing with implicit models. More precisely, we need to modify the construction of the Büchi automaton $\mathcal{A}_{M,q,A}$ that is used to accept the (q, A)-execution trees. Before, we simply checked all the moves of A in polynomial time and calculated the set of states A is effective for (as the moves are bounded by the number of transitions). Here, we have to incrementally generate all these moves from A using $\hat{o}$. This may take exponential time (as there can be exponentially many moves in terms of the number of states and agents). However, as this can be done independently of the non-emptiness check, the overall runtime of the algorithm is still double exponential. □

A summary of complexity results for the alternative representation of transitions is presented in Figure 12.

5.6 Higher-Order Representations of Models

Explicit models of realistic systems are prohibitively large, both in their size of state spaces and numbers of transitions. Thus, for practical verification, the model of a system must be given in a more compact way, for instance by generating the state space as valuations of some discrete-valued attributes, and defining transitions through their pre- and postconditions. In this section, we summarize very briefly the results for such high-level representations of multi-agent systems (e.g., concurrent programs, reactive modules, modular interpreted systems etc.). It is easy to see that unfolding a compact representation to an explicit model involves usually an exponential blowup in its size. Consider, for example, a system whose state space is defined by r boolean variables (binary attributes). Obviously, the number of global states in the system is $n = 2^r$. A more general approach is presented in [63], where the "high-level description" is defined in terms of *concurrent programs*, that can be used for simulating Boolean variables, but also for processes or agents acting in parallel.

A concurrent program P is composed of k concurrent processes, each described by a labeled transition system $P_i = \langle St_i, Act_i, \rightarrow_i, \mathcal{PV}_i, V_i \rangle$, where St_i is the set of local states of process i, Act_i is the set of local actions, $\rightarrow_i \subseteq St_i \times Act_i \times St_i$ is a local transition relation, and $\mathcal{PV}_i, V_i$ are the set of local propositions and their valuation. The behavior of program P is given by the product automaton of $P_1, \ldots, P_k$ under the assumption that processes work asynchronously, actions are interleaved, and synchronization is obtained through common action names.

Theorem 24 ([63]). *Model checking CTL in concurrent programs is* **PSPACE***-complete with respect to the number of local states and agents (processes), and the length of the formula.*

Analogous results have been proved for compact representations of repeated games.

Theorem 25 ([96, 48]). *Model checking ATL_{Ir} and ATL_{IR} in simple reactive modules [96] and modular interpreted systems [50, 48] is* **EXPTIME***-complete with respect to the number of local states and agents, and the length of the formula.*

The real surprise, however, comes to light when we study the model checking complexity for imperfect information agents.

Theorem 26 ([50, 48]). *Model checking ATL_{ir} in modular interpreted systems is* **PSPACE***-complete with respect to the number of local states and agents, and the length of the formula.*

Thus, model checking in modular interpreted systems seems to be easier for imperfect rather than perfect information strategies (while it appears to be distinctly harder for explicit models, cf. Sections 5.1-5.2). There are two reasons for that. The more immediate is that agents with limited information

have *fewer* available strategies than if they had perfect information about the current (global) state of the game. Generally, the difference is exponential in the number of agents. More precisely, the number of perfect information strategies is *double exponential* with respect to the number of agents and their local states, while there are "only" exponentially many uniform strategies – and that settles the results in favor of imperfect information.

The other reason is more methodological. While model checking imperfect information is easier when we are given a particular MIS, modular interpreted systems may provide more compact representation to systems where all the agents have perfect information by definition. In particular, the most compact MIS representation of a given CEGS M can be exponentially larger than the most compact MIS representation of M with the epistemic relations removed. This is because in the former case, the MIS must encode the epistemic relations explicitly (like in standard interpreted systems where epistemic relations are generated by local state spaces, cf. Section 2.5). In the latter case, the epistemic aspect is ignored, which gives some extra room for encoding the transition relation more efficiently.

On the other hand, it should be noted that for systems of agents with "reasonably imperfect information", i.e., ones where the number of each agent's local states is logarithmic in the number of global states of the system, the optimal MIS encodings for perfect and imperfect information are the same. Still, model checking ATL_{IR} is **EXPTIME**-complete and model checking ATL_{ir} is **PSPACE**-complete, which suggests that imperfect information can be beneficial in practical verification.

Finally, we report two results that are straightforward extensions of Theorem 20 and Theorem 26, respectively.

Theorem 27. *Model checking CL_{IR}, CL_{Ir}, CL_{ir}, and CL_{iR} is $\mathbf{\Delta_3^P}$-complete with respect to the number of local states and agents in the modular interpreted system and the length of the formula. Moreover, it is $\mathbf{\Sigma_2^P}$-complete for the "simple" variants of CL.*

Theorem 28. *Model checking ATL^*_{ir} in modular interpreted systems is* **PSPACE***-complete with respect to the number of local states and agents, and the length of the formula.*

Despite the pessimistic complexity results, several techniques have been proposed for practical model checking of knowledge, time, and strategies. We discuss them in the next section.

References: Our exposition of verification complexity for different variants of ATL follows mostly [9], though it was revised and updated with new results from e.g. [21, 11].

6 Practical Model Checking

In this section we discuss how model checking of CTLK can be made more feasible. We present approaches based on a translation to *Ordered Binary Decision Diagrams*, as well as ones that use a translation to Boolean satisfiability (so called *Bounded Model Checking* and *Unbounded Model Checking*). Finally, we look at Unbounded Model Checking for ATEL. We start with a short introduction to OBDDs and SAT solving.

6.1 Introduction to OBDDs

OBDDs (Ordered Binary Decision Diagrams) are used for succinct representation of Boolean functions. Model checking problem for CTLK can be efficiently encoded into operations on OBDDs. Consider a Boolean function:

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

Such a function can be represented by the results of all the valuations of some propositional formula over n propositional variables. For example the function $f(x_1, x_2) = x_1 * x_2$ is represented by the formula $p_1 \wedge p_2$. Each Boolean function can be represented by an OBDD. The size of the BDD is determined both by the function being represented and the chosen ordering of the variables. For a boolean function $f(x_1, \ldots, x_n)$ then depending upon the ordering of the variables we would end up getting a graph whose number of nodes would be linear (in n) at the best and exponential at the worst case.

Let us consider the Boolean function $f(x_1, \ldots, x_{2n}) = x_1x_2 + x_3x_4 + \cdots + x_{2n-1}x_{2n}$. Using the variable ordering $x_1 < x_3 < \cdots < x_{2n-1} < x_2 < x_4 < \cdots < x_{2n}$, the BDD needs 2^{n+1}, nodes to represent the function. Using the ordering $x_1 < x_2 < x_3 < x_4 < \cdots < x_{2n-1} < x_{2n}$, the BDD consists of $2n$ nodes.

The following operations can be implemented by polynomial-time graph manipulation algorithms: disjunction, conjunction, negation, implication, equivalence, existential abstraction, and universal abstraction.

6.2 Introduction to SAT

This part of the section aims at explaining the main principles followed by propositional SAT-solvers, i.e., algorithms testing satisfiability of propositional formulas. Our presentation is based on [6, 70].

Assume we are given a propositional formula φ. The aim of a SAT-solver is to find a satisfying assignment for φ if it exists, or return "unsatisfiable" otherwise. It is well known that the problem of establishing whether a formula is satisfiable or not (known as a SAT-problem) is NP-complete. Therefore, in general, one cannot expect that a SAT-solver will return a result in polynomial time. We should be aware of the fact that a SAT-solver is heuristics only, but it can be very "clever". Modern SAT-solvers can decide formulas composed of hundreds of

thousands of propositional variables in a reasonable time. Typically, SAT-solvers accept formulas in conjunctive normal form (CNF), i.e., a conjunction of clauses, where a *clause* is a disjunction of literals. Such a form is quite useful for checking satisfiability, as any valuation, which makes at least one literal of each clause satisfied, makes the whole formula satisfied.

Every propositional formula φ can be translated to a CNF formula in two ways. Either the resulting CNF formula preserves only satisfiability of φ or it is logically equivalent to φ. Clearly, the former translation is much easier than the latter and it is used for checking satisfiability of φ. If we need to operate further on φ, then an equivalence-preserving translation is necessary.

There are several approaches used to check satisfiability of propositional formulas. They can be based on Stålmarck's method [89], use methods of soft computing (Monte Carlo, evolutionary algorithms) or exploit the theory of resolution [30]. Here, we discuss the algorithm proposed by Davis and Putnam [18] and later improved by Davis, Logemann and Loveland [17], known as DPLL. The solution is based on a backtracking search algorithm through the space of possible assignments of a CNF formula. The algorithm uses the methods of *boolean constraint propagation* (*BCP*), *conflict-based learning* (*CBL*), and *variable selection* (*VS*).

References. Our presentation is based on [6, 70] and [76].

6.3 OBBD-Based Model Checking for CTLK

The algorithms from Section 3 (computing, for each formula φ, the set of states $[\![\varphi]\!]$ in which φ holds) can operate on the OBDD representations of the states. This requires to encode the states and the transition relation of a model M by propositional formulas, and then to represent these formulas by OBDDs.

The model checking problem $M, s^0 \models \varphi$ is translated to checking whether $s^0 \in [\![\varphi]\!]$. So, we need to verify whether $OBDD(\{s^0\}) \wedge OBDD([\![\varphi]\!])$ is not equal to $OBDD(\emptyset)$, where $OBDD(S)$ denotes the OBDD representing the set of states S.

6.4 Unbounded versus Bounded Model Checking

So far we have discussed model checking approaches that consist in checking a given formula over the whole model. We will refer to a symbolic version of this approach as to the *unbounded model checking* (UMC, for short). When the model we are dealing with is too large to be (even) symbolically encoded, one can apply the alternative approach, called *bounded model checking* (BMC, for short). In this case only an existential subset of the logic can be verified. BMC consists in translating the model checking problem of an existential modal formula (a formula containing only existential modalities) into the problem of satisfiability of a propositional formula[6]. In particular, the BMC algorithm checks for a finite witness among all (possibly infinite) paths of the system satisfying a given existential modal formula. Below, we discuss the above approaches in more detail.

[6] Alternativley, a BDD-based approach can be used

6.5 SAT-Based Approaches to Model Checking for CTLK

We extend CTLK with past operators to the logic CTLpK. The reason is twofold: to extend expressivness and to easily define unbounded model checking for CTLK.

- AHφ - always in the past φ holds,
- AYφ - for all the predecessor states φ holds,
- EPφ - for some state in the the past φ holds,
- EYφ - for some predecessor state φ holds.

The logic ACTLpK is the restriction of CTLpK such that it consists of the formulas of the form: AXα, AYα, A(αUβ), AGα, AHα, K$_i\alpha$, E$_\mathrm{A}\alpha$, D$_\mathrm{A}\alpha$, C$_\mathrm{A}\alpha$. So, the formulas are only in the universal form (no negation applied to modalities). The language of ECTLpK is defined as: $\{\neg\varphi \mid \varphi \in \text{ACTLpK}\}$.

After 'pushing' negation down the formula, we have the formulas only in the existential form (no negation applied to modalities): EXα, EYα, E(αUβ), EGα, EPα, $\overline{\mathrm{E}}_\mathrm{A}\alpha$, $\overline{\mathrm{D}}_\mathrm{A}\alpha$, $\overline{\mathrm{C}}_\mathrm{A}\alpha$.

Idea of Bounded Model Checking (BMC). The idea consists in proving that an ECTLpK formula holds or that an ACTLpK formula does not hold in M. Consider a formula φ. If $\varphi \in$ ACTLpK, then as φ we take its negation. So, we can assume that $\varphi \in$ ECTLpK. Next, we consider a fragment M' of the model M, preserving φ, i.e., such that $M' \models \varphi$ implies $M \models \varphi$. Consequently, we translate the model checking problem $M' \models \varphi$ to the problem of satisfiability of the propositional formula $[M'] \wedge [\varphi]_{M'}$, where $[M']$ is a propositional encoding of M' and $[\varphi]_{M'}$ is a propositional encoding of φ interpreted in M'. So, we have $M' \models \varphi$ iff $[M'] \wedge [\varphi]_{M'}$ is satisfiable. Next, we check the satisfiability of $[M'] \wedge [\varphi]_{M'}$ with a SAT-solver. If $[M'] \wedge [\varphi]_{M'}$ is satisfiable, then $M \models \varphi$.

Algorithm BMC for an ECTLpK formula φ. Typically, a fragment M' of the full model M is taken as an unfolding of M up to some depth k, called the k-model. Below, we define the k-model and the function $l$oop, which is used to check whether a finite path of length k (called a k-path) is a loop, i.e., represents an infinite path.

Definition 1 ($k-$model). *Let $M = (St, R, \sim_1, \ldots, \sim_n, V)$ be a model and $\iota \in St$ be the initial state. For simplicity, we assume that all the states St are reachable from ι, but this assumption can easily be dropped. The $k-$model for M is a structure $M_k = (St, P_k, \sim_1, \ldots, \sim_n, V)$, where St is the set of the global states, and P_k with $k \in \mathbb{N}_+$ is the set of all the k-paths of M. The function loop: $P_k \to 2^{\mathbb{N}}$ is defined as follows: loop$(\lambda) = \{l \mid 0 \leq l \leq k$ and $(\lambda(k), \lambda(l)) \in \to\}$.*

Satisfaction for the temporal formulas EGα in the bounded case depends on whether or not the k-computation λ defines a loop, i.e., whether *loop*$(\lambda) \neq \emptyset$.

Bounded semantics for ECTLpK. A propositional translation of an ECTLpK formula is based on the bounded semantics, which uses k-paths instead of infinite paths like in the standard semantics. Let $s \in St$ and M_k be the k-model for some $k \in \mathbb{N}_+$.

$s \models \mathrm{EX}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = s$ and $\lambda(1) \models \alpha$,
$s \models \mathrm{EG}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = s$ and $\forall_{0\le j\le k}\lambda(j) \models \alpha \wedge \mathit{loop}(\lambda) \neq \emptyset$,
$s \models \mathrm{E}(\alpha \mathrm{U} \beta)$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = s$ and $\exists_{0\le j\le k}\big(\lambda(j) \models \beta$ and $\forall_{0\le i<j}\lambda(i) \models \alpha\big)$.
$s \models \mathrm{EY}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(k) = s$ and $\lambda(k-1) \models \alpha$,
$s \models \mathrm{EP}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(k) = s$ and $\exists_{0\le j\le k}\lambda(j) \models \alpha$,
$s \models \overline{\mathrm{K}}_i\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = \iota$ and $\exists_{0\le j\le k}(s \sim_i \lambda(j)$ and $\lambda(j) \models \alpha)$,
$s \models \overline{\mathrm{E}}_\mathrm{A}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = \iota$ and $\exists_{0\le j\le k}\big((\exists i \in \mathrm{A})\ l_i(s) = l_i(\lambda(j))$ and $\lambda(j) \models \alpha\big)$,
$s \models \overline{\mathrm{D}}_\mathrm{A}\alpha$ iff there is a k-path $\lambda \in P_k$ s.t. $\lambda(0) = \iota$ and $\exists_{0\le j\le k}\big((\forall i \in \mathrm{A})\ l_i(s) = l_i(\lambda(j))$ and $\lambda(j) \models \alpha\big)$,
$s \models \overline{\mathrm{C}}_\mathrm{A}\alpha$ iff $s \models \bigvee_{i\le k}(\overline{\mathrm{E}}_\mathrm{A})^i\alpha$.

Intuitively, the bounded semantics for $s \models \overline{\mathrm{K}}_i\alpha$ says that there is a k-path λ, which starts at the beginning state ι, one of its states satisfies α and shares the same i-local state with s.
Several translations into boolean formulas can be defined that are directly based on the bounded semantics.

Computing the size of submodels of M_k. The function f_k is used for computing a sufficient number of k-paths in a submodel of M_k over which the formula is to be checked. Define the function $f_k : \mathrm{ECTLpK} \to \mathbb{N}$ as follows:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(Z\alpha) = f_k(\alpha) + 1$, for $Z \in \{\mathrm{EX}, \mathrm{EY}, \mathrm{EP}, \overline{\mathrm{K}}_i, \overline{\mathrm{D}}_\mathrm{A}, \overline{\mathrm{E}}_\mathrm{A}\}$,
- $f_k(\overline{\mathrm{C}}_\mathrm{A}\alpha) = f_k(\alpha) + k$,
- $f_k(\mathrm{EG}\alpha) = (k+1) \cdot f_k(\alpha) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$.

BMC Algorithm for ECTLpK.

- Let φ be an ECTLpK formula,
- Iterate for $k := 1$ to $|M|$,
- Select the $k-$model M_k (of the k-paths),
- Select the $f_k(\varphi)$-submodels of M_k (of $f_k(\varphi)$ k-paths),
- Translate the transition relation of the $k-$paths of M_k to a propositional formula $[M^{\varphi,\iota}]_k$, where ι is the initial state,
- Translate φ over all the $f_k(\varphi)$-submodels to a propositional formula $[\varphi]_{M_k}$,
- Check the satisfiability of $[M, \varphi]_k := [M^{\varphi,\iota}]_k \wedge [\varphi]_{M_k}$.

Unbounded Model Checking (UMC) for CTLpK. The method of UMC differs from BMC in the encoding of the formulas, while it shares with BMC the encoding of the states and the transition relation of the model. UMC exploits the characterisation of the basic modalities in Quantified Boolean Formulas (QBF) and algorithms that translate QBF and fixed point equations over QBF to propositional formulas.

$$M \models \varphi \text{ iff } [\varphi](w) \wedge I_\iota(w) \text{ is satisfiable.}$$

- w - a vector of propositional variables to encode the global states,
- $[\varphi](w)$ - a propositional encoding of the states of M in which φ holds,
- $I_\iota(w)$ - a symbolic encoding of the initial state ι.

In what follows $R(u, v)$ encodes the next step relation R, while $R_i(u, v)$ encodes the relation $\sim_i$, where u, v are vectors of propositional variables used for encoding the global states. UMC exploits three basic procedures:

1. $forall(v, \cdot)$ is applied to formulas $Z\alpha$ s.t. $Z \in \{\mathrm{AX}, \mathrm{AY}, \mathrm{K}_i, \mathrm{D}_\mathrm{A}, \mathrm{E}_\mathrm{A}\}$, where as before v is a vector of propositional variables to encode the global states. This procedure eliminates the universal quantifier from a QBF formula representing the formula $Z\alpha$ and returns the result in CNF.
2. $gfp_Z(\alpha)$ is applied to formulas $\mathrm{Z}\alpha$ s.t. $\mathrm{Z} \in \{\mathrm{AG}, \mathrm{AH}, \mathrm{C}_\mathrm{A}\}$.
3. $lfp_\mathrm{U}(\alpha, \beta)$ is applied to formulas $\mathrm{A}(\alpha \mathrm{U} \beta)$.

For 2 and 3 the procedures use the fixed point characterisation of the formulas. Operationally, one works outward from the most nested sub-formulas. To compute $[Z\alpha](w)$, where Z is a modality, we work under the assumption of already having computed $[\alpha](w)$. The formula $[\mathrm{AX}\alpha](w)$ is equivalent to the QBF formula $\forall v.(R(w, v) \Rightarrow [\alpha](v))$. Similar equivalences can be obtained for the formulas $\mathrm{AY}\alpha, \mathrm{K}_i\alpha, \mathrm{D}_\mathrm{A}\alpha$, and $\mathrm{E}_\mathrm{A}\alpha$. To calculate the actual translations we use either the fixed point or the QBF characterisation of the CTLpK formulas together with the three basic algorithms $forall()$, $gfp()$, and $lfp()$.

Procedure forall(). This procdeure eliminates the universal quantifiers from a QBF formula representing a modal formula. It is based on the Davis-Putnam-Logemann-Loveland approach.

Procedures gfp - the simplest versions (no optimizations).

procedure $gfp_{AG}([\alpha](w))$, `(* where` α `is a CTLpK formula *)`
$Q(w) := [true](w)$, $Z(w) := [\alpha](w)$
`while` $\neg(Q(w) \Rightarrow Z(w))$ `is satisfiable`
    $Q(w) := Z(w)$,
    $Z(w) :=$*forall*$(v, (R(w, v) \Rightarrow Z(v))) \wedge [\alpha](w)$
`return` $Q(w)$

procedure $gfp_{C_{\mathrm{A}}}([\alpha](w))$, (* where α is an CTLpK formula *)
$Q(w) := [true](w)$,
$Z(w) = forall\big(v,\ ((\bigvee_{i\in \mathrm{A}} R_i(w,v)) \Rightarrow [\alpha](v))\big)$;
while $\neg(Q(w) \Rightarrow Z(w))$ is satisfiable
    $Q(w) := Z(w)$,
    $Z(w) = forall(v,\ ((\bigvee_{i\in \mathrm{A}} R_i(w,v)) \Rightarrow (Z(v) \wedge [\alpha](v))))$,
return $Q(w)$

Procedure lfp - the simplest version (no optimizations).

procedure $lfp_{AU}([\alpha](w), [\beta](w))$, (* where α, β are CTLpK formulas *)
$Q(w) := [\mathbf{false}](w)$, $Z(w) := [\beta](w)$
while $\neg(Z(w) \Rightarrow Q(w))$ is satisfiable
    $Q(w) := Q(w) \vee Z(w)$,
    $Z(w) :=forall(v, (R(w,v) \Rightarrow Q(v))) \wedge [\alpha](w)$
return $Q(w)$

6.6 UMC for ATEL

Unbounded Model Checking can be also defined for ATL and, in fact, for the straightforward combination of ATL and the epistemic logic (ATEL). Let us assume that all the actions are represented in terms of their *pre* and *post* conditions, i.e., $pre(a)$ is a set of all the states from which the action a can be executed and $post(a)$ is a set of all states which can be reached after the execution of a. This means that the action a can be executed at any state in $pre(a)$ and takes to any state in $post(a)$. Moreover, by Act_i we mean the set of all the actions available to the agent i, where $i \in \mathbb{A}\mathrm{gt}$.

Next, we assume $St \subseteq \{0,1\}^m$, where $m = \lceil \log_2(|St|) \rceil$, i.e., every state is represented by a sequence consisting of 0's and 1's. Let Π be a set of fresh propositional variables such that $\Pi \cap \mathcal{PV} = \emptyset$, F_Π be a set of propositional formulas over Π, and $lit : \{0,1\} \times \Pi \to \mathrm{F}_\Pi$ be a function defined as follows: $lit(0,p) = \neg p$ and $lit(1,p) = p$. Furthermore, let $w = (w[1], \ldots, w[m])$ be a *global state variable*, where $w[i] \in \Pi$ for each $i = 1, \ldots, m$. We use elements of St as valuations[7] of global states variables in formulas of F_Π. For example $w[1] \wedge w[2]$ evaluates to $true$ for the valuation $q = (1, \ldots, 1)$, and it evaluates to $false$ for the valuation $q = (0, \ldots, 0)$.

Now, the idea consists in using propositional formulas of F_Π to encode sets of states of St. For example, the formula $w[1] \wedge \ldots \wedge w[m]$ encodes the state represented by $(1, \ldots, 1)$, whereas the formula $w[1]$ encodes all the states, the first bit of which is equal to 1.

[7] We identify 1 with $true$ and 0 with $false$.

The following propositional formulas are defined:

- $I_q(w) := \bigwedge_{i=1}^{m} lit(q[i], w[i])$,

  this formula encodes the state $q = (q[1], \ldots, q[m])$, i.e., $q[i] = 1$ is encoded by $w[i]$, and $q[i] = 0$ is encoded by $\neg w[i]$; notice that $q = (q[1], \ldots, q[m])$ is the only valuation of the global state variable $w = (w[1], \ldots, w[m])$ that satisfies $I_q(w)$,

- $pre_a(w)$ and $post_a(w)$ for every $a \in \bigcup_{i \in \mathbb{A}\mathrm{gt}} Act_i$,

  $pre_a(w)$ is a formula which is true for a valuation $q = (q[1], \ldots, q[m])$ of $w = (w[1], \ldots, w[m])$ iff $q \in pre(a)$ and $post_a(w)$ is a formula which is true for a valuation $q = (q[1], \ldots, q[m])$ of $w = (w[1], \ldots, w[m])$ iff $q \in post(a)$.

The key element of the method is to translate ATL formulas into propositional formulas. Specifically, for a given ATL formula ϕ we compute a corresponding propositional formula $[\phi](w)$ which is satisfied by a valuation $q = (q[1], \ldots, q[m])$ of $w = (w[1], \ldots, w[m])$ iff $s \models \phi$. In so doing we obtain a formula $[\phi](w)$ such that ϕ is valid in the model iff the conjunction $[\phi](w) \wedge I_\iota(w)$ is satisfiable. Notice that $[\phi](w) \wedge I_\iota(w)$ is satisfiable only if $[\phi](w)$ is valid for the valuation implied by the initial state ι. Operationally, we work outwards from the most nested subformulas, i.e., to compute $[O\alpha](w)$, where O is a modality, we work under the assumption of already having computed $[\alpha](w)$.

Given an ATEL formula φ, the propositional translation $[\varphi](w)$ is inductively defined as follows:

- let $A = \{i_1, .., i_t\} \subseteq \mathbb{A}\mathrm{gt}$ and $B = \{j_1, .., j_q\} = \mathbb{A}\mathrm{gt} \backslash A$,
  $[\langle\langle A \rangle\rangle \mathrm{X}\alpha](w) := \bigvee_{c_{i_1} \in Act_{i_1}, .., c_{i_t} \in Act_{i_t}} \big(\bigwedge_{i'=1}^{t} pre_{c_{i_{i'}}}(w) \wedge forall\big(v, \bigwedge_{c_{j_1} \in Act_{j_1}}, .., c_{j_q} \in Act_{j_q} \big(\bigwedge_{j'=1}^{q} pre_{c_{j_{j'}}}(w) \wedge \bigwedge_{j=1}^{q} post_{c_{j_{j'}}}(v) \wedge \bigwedge_{i'=1}^{t} post_{c_{i_{i'}}}(v) \Rightarrow [\alpha](v)\big)\big)\big)\big)$,
- $[\mathrm{K}_i\alpha](w) := forall\big(v, (R_i(w, v) \Rightarrow [\alpha](v))\big)$,
- $[\mathrm{E}_A\alpha](w) := forall\big(v, ((\bigvee_{i \in A} R_i(w, v)) \Rightarrow [\alpha](v))\big)$,
- $[\mathrm{D}_A\alpha](w) := forall\big(v, ((\bigwedge_{i \in A} R_i(w, v)) \Rightarrow [\alpha](v))\big)$,
- $[\langle\langle A \rangle\rangle \mathrm{G}\alpha](w) := gfp_A([\alpha](w))$,
- $[\langle\langle A \rangle\rangle \alpha \mathrm{U} \beta](w) := lfp_A([\alpha](w), [\beta](w))$,
- $[\mathrm{C}_A\alpha](w) := gfp_{C_\mathrm{A}}([\alpha](w))$.

The algorithms gfp and lfp are based on the standard procedures computing fixed points and are given below. In order to simplify the subsequent notation, we use $[\langle\langle A \rangle\rangle \mathrm{X}Z(w)](w)$ to denote the formula: $\bigvee_{c_{i_1} \in Act_{i_1}, .., c_{i_t} \in Act_{i_t}} \big(\bigwedge_{i'=1}^{t} pre_{c_{i_{i'}}}(w) \wedge forall\big(v, \bigwedge_{c_{j_1} \in Act_{j_1}}, .., c_{j_q} \in Act_{j_q} \big(\bigwedge_{j'=1}^{q} pre_{c_{j_{j'}}}(w) \wedge \bigwedge_{j=1}^{q} post_{c_{j_{j'}}}(v) \wedge \bigwedge_{i'=1}^{t} post_{c_{i_{i'}}}(v) \Rightarrow Z(v)\big)\big)\big)$, where $Z(v)$ is a propositional formula over the global variable v encoding a subset of St.

procedure $gfp_A([\alpha](w))$, `(* where` α `is an ATEL formula *)`

$Q(w) := [true](w)$, $Z(w) := [\alpha](w)$
`while` $\neg(Q(w) \Rightarrow Z(w))$ `is satisfiable`
    $Q(w) := Z(w)$,
    $Z(w) := [\langle\langle A\rangle\rangle \mathrm{X} Z(w)](w) \wedge [\alpha](w)$
`return` $Q(w)$

procedure $lfp_A([\alpha](w), [\beta](w))$, `(*where` α, β `are ATEL formulas *)`
$Q(w) := [false](w)$, $Z(w) := [\beta](w)$
`while` $\neg(Z(w) \Rightarrow Q(w))$ `is satisfiable`
    $Q(w) := Q(w) \vee Z(w)$,
    $Z(w) := [\langle\langle A\rangle\rangle \mathrm{X}\, Q(w)](w) \wedge [\alpha](w)$
`return` $Q(w)$

References: Our presentation is based on the following papers:

- OBBD approach for CTLK - [83, 69],
- BMC approach for CTLK - [74, 75],
- BMC approach for CTLpK - [58],
- UMC approach for CTLK - [59, 56, 57],
- UMC approach for ATEL - [61].

For further reading, consult [68]. More details about the `VerICS` model checker can be found in [20, 60].

7 Final Remarks

In these materials we surveyed several important logics for specification of multi-agent systems, and the related approaches to automatic verification for MAS. We paired specification languages with complexity results on their model checking problems with different representations of systems. Moreover, we discussed several model checking algorithms for temporal-epistemic and strategic logics, starting from ones working directly on Kripke models, to those exploiting operations on Binary Decision Diagrams and efficient SAT-solvers. Bounded and unbounded model checking approaches have been identified and discussed as well.

We would like to emphasize that specification and verification of multi agent systems is a very rapidly developing area of computer science. Therefore, even at the time of writing these materials new methods are emerging. In order to become acquainted with them we refer the interested reader to the proceedings of conferences AAMAS, IJCAI, ICSOC, ICEFM, MoChart, ATVA, etc. Below we hesitantly list some open problems within the area, expecting that – at the time this chapter gets to the reader – some of them may have been already solved, and certainly many more research questions will have emerged:

- Logics for specification of MAS with perfect information are relatively well studied. However, many issues regarding modeling, analysis, and reasoning about systems with incomplete information are barely touched. This regards e.g. axiomatization of ATL with imperfect information (some preliminary results have been obtained in [35] but the research is far from complete). Moreover, to our best knowledge, no satisfiability-checking algorithms for any variant of ATL with imperfect information have been proposed;
- Model checking of strategies with incomplete information is largely left untouched (besides the theoretical complexity results that we presented in Section 5);
- The coordination issue: the ATL formula $\langle\langle A\rangle\rangle\gamma$ only requires that there *exists* a winning strategy for A to achieve γ. However, the agents in A may not be able to successfully coordinate if there are multiple joint strategies with that property. Some work on this issue has been reported in [40] but the research is still preliminary;
- Correspondence between abstract and concrete models of agents' strategic play is becoming to attract interest again. The issue is relatively well studied for one-step games (cf. [73, 33]) but only begun to be investigated for multi-step games where paths rather than states are outcomes [32];
- The connection between strategic logics and mainstream game theory is still weak in the sense that game theory uses much more sophisticated models of agents' incentives and concepts of agents' rationality. There have been several attempts at characterizing game-theoretic solution concepts in modal logics (e.g., [39, 38, 95, 103, 12]) but none of them satisfactory. In particular, none of the proposals match the elegance and simplicity of models and concept definitions in game theory;

– There are also open problems more closely related to practical verification of multi-agent systems. Mainly, they concern applications of methods known for verification of temporal logics, but not yet for logics of MAS. These are bounded model checking for ATL, a combination of SAT- and BDD-based verification methods for CTLK and ATL, verification of parametric versions of epistemic and strategy logics and symbolic verification of hybrid and probabilistic real-time epistemic logics.

References

1. T. Agotnes. Action and knowledge in alternating-time temporal logic. *Synthese*, 149(2):377–409, 2006. Section on Knowledge, Rationality and Action.
2. T. Agotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logics with irrevocable strategies. In D. Samet, editor, *Proceedings of TARK XI*, pages 15–24, 2007.
3. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.
4. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.
5. C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259, 1980.
6. A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In *Highly Dependable Software*, volume 58 of *Advances in Computers*. Academic Press, 2003. Pre-print.
7. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
8. N. Bulling. Model checking coalition logic on implicit models is Δ_3-complete. Technical Report IfI-10-02, Clausthal University of Technology, 2010.
9. N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.
10. N. Bulling and W. Jamroga. What agents can probably enforce. *Fundamenta Informaticae*, 93(1-3):81–96, 2009.
11. N. Bulling and W. Jamroga. Alternating epistemic mu-calculus. In *Proceedings of IJCAI-11*, pages 109–114, 2011.
12. N. Bulling, W. Jamroga, and J. Dix. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):51–114, 2008.
13. E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
14. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
15. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
16. P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
17. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7):394–397, 1962.
18. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
19. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345:139–170, 2005.
20. P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Polrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS: A tool for verifying timed automata and

Estelle specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 278–283. Springer-Verlag, 2003.
21. C. Dima and F.L. Tiplea. Model-checking atl under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.
22. L. Doyen and J.-F. Raskin. Games with imperfect information: Theory and algorithms. In *Lecture Notes in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press, 2011.
23. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.
24. E. A. Emerson and C-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. of the 1st Symp. on Logic in Computer Science (LICS'86)*, pages 267–278. IEEE Computer Society, 1986.
25. E. A. Emerson and Ch.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.
26. E. A. Emerson and A. P. Sistla. Deciding branching time logic. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 14–24, New York, NY, USA, 1984. ACM.
27. E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.
28. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
29. M. Fisher. *Temporal Logics*. Kluwer, 2006.
30. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.
31. M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.
32. V. Goranko and W. Jamroga. State and path effectivity models for logics of multi-player games. In *Proceedings of AAMAS*, 2012. To appear.
33. V. Goranko, W. Jamroga, and P. Turrini. Strategic games and truly playable effectivity functions. In *Proceedings of AAMAS2011*, pages 727–734, 2011.
34. V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1):93–117, 2006.
35. D.P. Guelev, C. Dima, and C. Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.
36. J. Y. Halpern. Reasoning about knowledge: a survey. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume IV*, pages 1–34. Oxford University Press, 1995.
37. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
38. B.P. Harrenstein, W. van der Hoek, J.-J. Meyer, and C. Witteveen. A modal characterization of Nash equilibrium. *Fundamenta Informaticae*, 57(2–4):281–321, 2003.
39. P. Harrenstein, W. van der Hoek, J-J. Meijer, and C. Witteveen. Subgame-perfect Nash equilibria in dynamic logic. In M. Pauly and A. Baltag, editors, *Proceedings of the ILLC Workshop on Logic and Games*, pages 29–30. University of Amsterdam, 2002. Tech. Report PP-1999-25.

40. P. Hawke. Coordination, almost perfect information and strategic ability. In *Proceedings of LAMAS*, 2010.
41. W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.
42. X. Huang, K. Su, and C. Zhang. Probabilistic alternating-time temporal logic of incomplete information and synchronous perfect recall. In *Proceedings of AAAI-12*, 2012. To appear.
43. M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.
44. N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
45. W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Keplicz and R. Verbrugge, editors, *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003)*, pages 133–140, 2003.
46. W. Jamroga. A temporal logic for stochastic multi-agent systems. In *Proceedings of PRIMA'08*, volume 5357 of *LNCS*, pages 239–250, 2008.
47. W. Jamroga. Easy yet hard: Model checking strategies of agents. In *Computational Logic in Multi-Agent Systems: Proceedings of CLIMA IX*, volume 5405 of *LNCS*, pages 1–12, 2009.
48. W. Jamroga and T. Agotnes. Modular interpreted systems: A preliminary report. Technical Report IfI-06-15, Clausthal University of Technology, 2006.
49. W. Jamroga and T. Agotnes. Constructive knowledge: What agents can achieve under incomplete information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.
50. W. Jamroga and T. Agotnes. Modular interpreted systems. In *Proceedings of AAMAS'07*, pages 892–899, 2007.
51. W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pěchouček, P. Petta, and L.Z. Varga, editors, *Proceedings of CEEMAS 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer Verlag, 2005.
52. W. Jamroga and J. Dix. Model checking ATL_{ir} is indeed Δ_2^P-complete. In *Proceedings of EUMAS'06*, 2006.
53. W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 42(3):366–410, 2008.
54. W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.
55. G. Jonker. Feasible strategies in Alternating-time Temporal Epistemic Logic. Master thesis, University of Utrecht, 2003.
56. M. Kacprzak, A. Lomuscio, T. Lasica, W. Penczek, and M. Szreter. Verifying multi-agent systems via unbounded model checking. In *FAABS*, pages 189–212, 2004.
57. M. Kacprzak, A. Lomuscio, A. Niewiadomski, W. Penczek, F. Raimondi, and M. Szreter. Comparing BDD and SAT based techniques for model checking Chaum's dining cryptographers protocol. *Fundamenta Informaticae*, 72(1-2):215–234, 2006.
58. M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundamenta Informaticae*, 63(2-3):221–240, 2004.
59. M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundam. Inform.*, 63(2-3):221–240, 2004.

60. M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae*, 85(1-4):313–328, 2008.
61. M. Kacprzak and W. Penczek. Fully symbolic unbounded model checking for alternating-time temporal logic. *Autonomous Agents and Multi-Agent Systems*, 11(1):69–89, 2005.
62. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
63. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
64. F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4:7, 2008.
65. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTL+ and FCTL is hard. In *Proceedings of FoSSaCS'01*, volume 2030 of *Lecture Notes in Computer Science*, pages 318–331. Springer, 2001.
66. K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool, 2008.
67. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 97–107, New York, NY, USA, 1985. ACM.
68. A. Lomuscio and W. Penczek. Logic column 19: Symbolic model checking for temporal-epistemic logics. *CoRR*, abs/0709.0446, 2007.
69. A. Lomuscio, H. Qu, and F. Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *CAV*, pages 682–688, 2009.
70. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. of the 38th Design Automation Conference (DAC'01)*, pages 530–535, June 2001.
71. M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
72. C.H. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.
73. M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001.
74. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. In *Proc. of the 2nd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, pages 209–216. ACM, 2003.
75. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.
76. W. Penczek and A. Polrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*. Springer-Verlag, 2006.
77. G. Peterson and J. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 348–363. IEEE Computer Society Press, 1979.
78. G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers and Mathematics with Applications*, 41(7):957–992.
79. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, pages 179–190, New York, NY, USA, 1989. ACM.

80. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 746–757. IEEE Computer Society Press, 1990.
81. F. Raimondi. *Model Checking Multi-Agent Systems*. PhD thesis, University College London, 2006.
82. F. Raimondi and A. Lomuscio. Automatic verification of deontic interpreted systems by model checking via OBDD's. In R.L. de Mántaras and L. Saitta, editors, *Proceedings of ECAI*, pages 53–57, 2004.
83. F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *AAMAS*, pages 630–637, 2004.
84. A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.
85. R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, 1992.
86. Ph. Schnoebelen. The complexity of temporal model checking. In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific, 2003.
87. Henning Schnoor. Strategic planning for probabilistic games with incomplete information. In *Proceedings of AAMAS'10*, pages 1057–1064, 2010.
88. P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.
89. M. Sheeran, S. Singh, and G. Stålmarck. Checking safety properties using induction and a SAT-solver. In *Proc. of the Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD'00)*, volume 1954 of *LNCS*, pages 108–125. Springer-Verlag, 2000.
90. N. V. Shilov, N. O. Garanina, and K.-M. Choe. Update and abstraction in model checking of knowledge and branching time. *Fundamenta Informaticae*, 72(1-3):347–361, 2006.
91. N. V. Shilov, N. O. Garanina, and N. A. Kalinina. Model checking knowledge, actions and fixpoints. In *Proceedings of CS&P2004*, pages 351–357, 2004.
92. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of ACM*, 32(3):733–749, 1985.
93. C. Stirling. *Modal and Temporal Properties of Processes*. Springer-Verlag, 2001.
94. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
95. W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of AAMAS'05*, pages 157–164, 2005.
96. W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In P. Stone and G. Weiss, editors, *Proceedings of AAMAS'06*, pages 201–208, 2006.
97. W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. *Game Theory and Applications*, 8:53–94, 2002.
98. W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 1167–1174. ACM Press, New York, 2002.
99. W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time Temporal Epistemic Logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

100. R. van der Meyden and N.V. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *LNCS*, pages 432–445. Springer, 1999.
101. S. van Otterloo and G. Jonker. On Epistemic Temporal Strategic Logic. *Electronic Notes in Theoretical Computer Science*, XX:35–45, 2004. Proceedings of LCMAS'04.
102. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344. IEEE Computer Society Press, 1986.
103. D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In D. Samet, editor, *Proceedings TARK XI*, pages 269–278. Presses Universitaires de Louvain, 2007.
104. G. Weiss, editor. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press: Cambridge, Mass, 1999.
105. M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.